**ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA**

Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

# EleKtion: una Libreria Kotlin Multiplatform per la Democrazia Digitale in Nuovi Contesti

Tesi di laurea in:
Laboratorio Di Sistemi Software

*Relatore*
**Prof. Danilo Pianini**

*Candidato*
**Jacopo Corina**

IV Appello della Sessione di Laurea
Anno Accademico 2022-2023

# Sommario

La tesi si concentra sull'evoluzione della democrazia digitale e l'implementazione di EleKtion, una libreria multipiattaforma per votazioni. Dopo un' introduzione sulla definizione di democrazia, si analizzano le caratteristiche delle votazioni e si progetta un linguaggio di dominio volto a facilitare la definizione del contesto della votazione. La fase di implementazione si svolge affrontando aspetti cruciali come la gestione degli artefatti e l'implementazione dei casi di test. In seguito si sperimenta la libreria utilizzando dati reali delle competizioni di Formula 1, evidenziando come logiche algoritmiche personalizzabili possano generare esiti alternativi, confrontandoli infine con quelli reali per una valutazione qualitativa del sistema.

*Ai miei genitori*

# Indice

# Elenco delle figure

# Elenco degli estratti di codice

# Capitolo 1

# Introduzione

Nell'ultimo ventennio l'informatica sta perpetuando un progresso tecnologico sempre più pervasivo nella quotidianità delle persone, digitalizzando i processi in numerosi ambiti, come le procedure burocratiche, la razionalizzazione delle risorse, la raccolta di opinioni di una comunità, con la tendenza di giungere ad una semplificazione e ad una maggior diffusione sulle informazioni di interesse per gli *stakeholders*. Questo fenomeno ha impattato, pur con qualche difficoltà, anche uno dei processi che tradizionalmente è più legato ad una lavorazione analogica e ad una maggior attenzione da parte del pubblico, ossia il contesto delle votazioni e dell'applicazione del potere democratico all'interno di uno Stato. L'informatizzazione delle votazioni è stata adattata a molteplici contesti, dalle votazioni per le elezioni dei rappresentanti politici a decisioni interne tra gruppi di lavoro e consultazioni della volontà popolare. L'espansione potrà in futuro coinvolgere nuovi settori, come quello sportivo, ponendo sugli esiti una maggior influenza fondata su nuovi criteri e volontà popolari. Lo scopo della tesi è realizzare una libreria in Kotlin Multiplatform chiamata EleKtion, la quale permette di creare e gestire votazioni lasciando all'utilizzatore la possibilità di parametrizzazione. La disponibilità degli artefatti finali su molteplici piattaforme permette di massimizzarne la diffusione, garantendo allo stesso tempo che il prodotto sia stato generato seguendo criteri di qualità, sia a livello di logiche di funzionamento che correttezza sintattica del codice sorgente. In questa tesi viene presentata inizialmente una panoramica relativa

alla nascita della democrazia, di come essa è mutata nel tempo, e di come l'approccio informatico può, e in certi casi ha potuto, riportare maggior potere nella volontà popolare, abbattendo barriere geografiche e sociali. Successivamente viene mostrata l'analisi e la modellazione del dominio per la gestione delle votazioni, approfondendo le differenti tipologie di voto. Viene poi esposta la progettazione di un linguaggio di dominio utile a definire con maggior facilità il contesto della votazione, generando diagrammi UML architetturali, per poi vagliare gli aspetti implementantivi della libreria, partendo dagli strumenti che semplificano l'evoluzione e la pubblicazione del software fino ad arrivare ai casi di test. Infine, viene illustrata una sperimentazione della libreria utilizzando dati relativi a campionati sportivi, nello specifico scegliendo come sport la Formula 1. L'applicazione di logiche algoritmiche, definibili dall'utilizzatore, e la possibilità di dichiarare elenchi di voti nelle gare, pongono nuovi scenari che possono modificare e mutare la storia di un campionato. Questi risultati vengono confrontati con quelli reali per ottenere un'evidenza utile ad una valutazione qualitativa.

Il termine *democrazia*, derivante dal greco *démos krátos* (governo del popolo), indica una forma di governo nella quale il potere appartiene al popolo ed è esercitato attraverso forme dirette, che coinvolgono attivamente i cittadini, e attraverso forme rappresentantive, che prevedono un sistema di rappresentanza rinnovata ciclicamente attraverso apposite elezioni. In una qualunque forma di *democrazia*, sono fondamentali i principi di elezioni libere, equità dei cittadini, partecipazione attiva alla vita politica, protezione dei diritti fondamentali e della libertà personale attraverso costituzioni [VK17]. Nelle epoche successive alle prime forme di democrazia "unanime" (per quanto il concetto di cittadinanza fosse assai diverso da quello attuale, ad esempio, erano escluse donne, schiavi e minori di 20 anni), complice la complessità delle materie da amministrare e il numero crescente di partecipanti fisici, la democrazia rappresentativa ha preso la maggior diffusione. La modernizzazione tecnologica accelerata di recente con l'avvento della pandemia da Covid-19, che ha evidenziato notevoli problemi logistici tra cui la gestione del distanziamento [Sot20], ha riacceso l'opportunità di permettere il ritorno ad un controllo più diretto da parte della collettività. La *democrazia digitale*

(o *e-democracy*) è definibile come "l'esercizio di processi decisionali democratici attraverso l'uso di informazioni e tecnologie abilitanti alla comunicazione, in particolare Internet" [RSW22]. Queste tecnologie, abilitanti a superare limitazioni territoriali e sociali, facilitano l'esercizio di ulteriori forme di democrazia, come quella partecipata, in cui i cittadini, piuttosto che sostituirsi ai rappresentanti, forniscono idee sullo sviluppo dell'indirizzo di governo permettendo così di creare un collettore di confronto ed analisi di situazioni disomogenee. Nonostante la virtualizzazione della partecipazione permetterebbe alle fasce più disagiate di essere maggiormente coinvolte, introduce allo stesso tempo un motivo di scetticismo relativo alla componente tecnologica in ambito di sicurezza informatica, usabilità, segretezza e manipolazione del voto [AR19]. Stabilire un legame di fiducia con queste tecnologie, normarne la verificabilità e pattuire principi di *accountability* sono i fattori principali e quindi pre-condizioni essenziali per una diffusione efficace. Alcuni autori sostengono che è impossibile avere leggitimità in quanto "la natura dei computer è tale che il loro funzionamento interno è segreto. Poiché le transazioni e i calcoli avvengono a livello elettronico, non è fisicamente possibile per gli esseri umani osservare esattamente ciò che un computer sta facendo" [MM04]. L'applicazione più comune di queste tecnologie emergenti è legata al mondo della politica [AR19]: sono degni di nota casi come l'Estonia, che dal 2005 rende possibili via Internet le votazioni per elezioni e referendum, e la Svizzera, che dal 2015 rende progressivamente disponibile la possibilità di votare online ai cittadini, sulla base delle regole operative che i Cantoni decidono seguendo le linee guida dettate dalla Confederazione Svizzera. In Italia possiamo rilevare l'applicativo PartecipaMi[1], organizzato per mettere in contatto la popolazione milanese con l'amministrazione comunale. Inoltre, queste soluzioni sono spesso adottate da partiti o movimenti politici per avviare iniziative e votazioni interne tra i membri, come nel caso del Partito Pirata tedesco in Germania (piattaforma LiquidFeedback), o il Movimento 5 Stelle in Italia (piattaforma Rousseau). Lo scopo principale è quello di promuovere la discussione di proposte tra i membri di un'organizzazione, le quali saranno portate in sede legislativa da un loro rappresentante [KLN19]. Sistemi come Li-

[1] `https://archive.is/HIXoC`

quidFeedback, OpenDCN, Airesis, essendo strutturati come applicativi end-to-end, pongono molta enfasi sull'interazione tra utenti, la presentazione e validazione di proposte, l'eventuale presenza di deleghe, l'espressione del voto tramite una lista di preferenze, ma non permettono la variazione delle logiche da applicare per la selezione del vincitore [Tra]. In altri sistemi, una fase di analisi iniziale poco trasversale ha portato a dar molto risalto ad aspetti non funzionali, come la privacy, e scarsità di approfondimento per quelli funzionali, come il modello di democrazia da applicare, che per alcuni aspetti rimane implicito [PO18]. Ciò è dovuto anche al fatto che il maggior coinvolgimento si concentra principalmente alle fasi iniziali e finali del ciclo della votazione [HKvK+19]. La mancanza di elementi formalizzanti, che descrivono dettagliatamente tutti gli step applicati nel processo, unita alle variazioni ed evoluzioni che vengono apportate nel tempo, portano alla divergenza del flusso rispetto all'idea iniziale, che conseguentemente non risulterà chiaro agli utilizzatori e potenzialmente potrebbe scoraggiare l'utilizzo del sistema [PO18].

Al di là dell'applicazione di *e-democracy* nel mondo della politica, che riscontra ostacoli legati più all'aspetto organizzativo ed umano che ad aspetti di evoluzione tecnologica, vi sono esempi notabili anche nel campo del *pervasive computing*, ad esempio nell'ambito delle *smart cities*. Ricollegandosi al fenomeno dei cambiamenti climatici che stanno inesorabilmente continuando ad avanzare, in certe zone del mondo l'acqua potabile sta diventando un bene sempre meno reperibile e degno di salvaguardia. In [BT17] si pone il focus sull'adozione di sistemi di *IdroInformatica*. Questo tipo di sistema permette di analizzare e monitorare, attraverso sensori e componenti real-time *SCADA* (Supervisory Control And Data Acquisition), gli attuali flussi e pressioni nella rete di distribuzione ed effettuare predizioni sui futuri guasti. Questi strumenti di analisi possono fornire ai cittadini indicazioni su come ottimizzare l'utilizzo delle risorse. L'unione di questi sistemi ad un coinvolgimento digitale e democratico, che porti i cittadini a prendere parte alle decisioni per i cambiamenti della gestione presente in una determinata area, permetterebbe una miglior localizzazione circa le soluzioni da intraprendere, ed ai cittadini una maggior consapevolezza, in quanto sono i principali *stakeholders*, oltre ad attribuirli maggior responsabilità. In altri Stati, come Canada e Paesi Bassi, la

gestione dell'acqua è delegata ad iniziative locali che promuovono la collaborazione tra soggetti pubblici e privati. Questi nuovi approcci alla gestione democratica degli interessi dei cittadini migliorano i rapporti di trasparenza e fiducia reciproca, permettendo una maggior capacità di espressione verso i rappresentanti ma anche da parte di chi ora è in grado di recepire, in ottica di miglioramento continuo, le esigenze del mondo "reale". L'intermediazione, applicata a questi nuovi contesti, continua a rimanere essenziale. Piuttosto che diventare un nuovo "medium" per imporre decisioni, risulta più efficace per favorire lo scambio costruttivo, mantenendo comunque le opportune autonomie riguardo alle decisioni finali, tutelando gli interessi collettivi e al contempo educando le persone a mantenere la libertà di pensiero [CF19].

# Capitolo 2

# Analisi

In questo capitolo viene descritta l'analisi sugli aspetti caratterizzanti la libreria multipiattaforma EleKtion. Viene esposto un approfondimento sui tipi di voto ed alcuni algoritmi applicabili per ciascun tipo; successivamente viene definito un elenco di requisiti fondamentali per le fasi successive e infine, viene fornita una modellazione del dominio mediante diagramma UML.

## 2.1 Votazioni a singola preferenza

La tipologia di voto a singola preferenza impone al votante di effettuare una scelta tra un numero definito di candidati. Al termine della votazione, tipicamente si adotta il criterio maggioritario: si contano i voti ottenuti da ciascun candidato, e colui che ottiene il numero più elevato risulta vincitore. A seconda del contesto preso in considerazione, è possibile avere più turni per diminuire di volta in volta il numero di candidati coinvolti. Se si arriva ad un pareggio, è possibile effettuare un nuovo ballottaggio oppure adottare criteri secondari, che variano in base al dominio trattato.

## 2.2 Votazioni a lista di preferenze

La tipologia di voto a lista di preferenze non impone al votante di effettuare un'unica scelta, bensì di fissare un ordinamento tra i candidati che parta dal minor favorito al maggior favorito o viceversa. Questa logica permette di conservare più contenuto informativo sulle preferenze rispetto alla singola scelta, e offre anche un dettaglio di gradimento degli altri candidati. Questo tipo di voto è stato introdotto inizialmente dall'algoritmo di Condorcet ed è stato adottato anche nelle sue varianti. Di seguito viene data una spiegazione sul funzionamento dell'algoritmo di Condorcet, e di un suo derivato, l'algoritmo di Schultze.

### 2.2.1 Algoritmo di Condorcet

L'algoritmo di Condorcet permette di estrarre un vincitore tra $n$ candidati, applicando il criterio di maggioranza tra le coppie di candidati. Si generano le possibili coppie, si confrontano e si valuta quale candidato batte tutti gli altri nello scontro di coppia. All'interno di una coppia, un candidato è vincitore se occupa una posizione preferenziale rispetto all'altro. Di seguito viene riportato un esempio, supponendo di avere tre possibili candidati, A, B, C, e 60 differenti votanti, che ordinano i candidati dalla posizione 1 alla posizione 3 dove la posizione 1 indica la posizione di maggior preferenza. Si ottengono delle liste di preferenze come rappresentato in tabella 2.1, in cui è mostrata l'occorrenza di ciascuna lista.

| Posizione 1 | Posizione 2 | Posizione 3 | Numero di occorrenze della lista |
|---|---|---|---|
| A | C | B | 23 |
| B | C | A | 19 |
| C | B | A | 16 |
| C | A | B | 2 |

Tabella 2.1: Tabella delle liste di preferenze, con l'occorrenza di ciascuna lista

Ora è possibile effettuare confronti tra le coppie, partendo ad esempio dalla coppia (B,C)

- Nella prima lista C è in una posizione favorevole rispetto a B, quindi C ottiene 23 punti
- Nella seconda lista B è in una posizione favorevole rispetto a C, quindi B ottiene 19 punti
- Nella terza lista C è in una posizione favorevole rispetto a B, quindi C ottiene 16 punti
- Nella quarta lista C è in una posizione favorevole rispetto a B, quindi C ottiene 2 punti

Si ha che il candidato C batte il candidato B con un punteggio di 41 a 19. Facendo i confronti tra le coppie (A,B) e (A,C) si ottiene che il candidato B batte il candidato A con un punteggio di 35 a 25, e il candidato C batte il candidato A con un punteggio di 37 a 23. Il risultato finale è quindi il seguente:

- Posizione 1: C
- Posizione 2: B
- Posizione 3: A

In alcuni casi il metodo di Condorcet può non portare ad alcun vincitore, presentando un ciclo e trovandosi, ad esempio, in una situazione in cui C batte B, B batte A, A batte C. Le varianti del metodo di Condorcet ottimizzano l'algoritmo per minimizzare il rischio di indeterminazione del vincitore. Può accedere, in Condorcet come nelle sue varianti, di ottenere dei pareggi, i quali avvengono quando due o più concorrenti pareggiano l'uno con l'altro ma battono tutti i restanti. La situazione di pareggio può essere gestita in vari modi, ad esempio, estraendo il vincitore in modo casuale, valutando altri criteri come la preferenza nei singoli voti (most first choice) oppure mantenendo il set di pareggio senza effettuare alcuna azione. Dato il risultato della votazione, è possibile definire l'utilità individuale come il grado di soddisfazione del votante. Il grado di soddisfazione sarà tanto più alto quanto più vicino è il risultato rispetto al voto espresso. Concludendo,

l'algoritmo porta ad un risultato nel quale si tende a massimizzare l'utilità per il maggior numero di votanti, ossia a massimizzare l'utilità complessiva.

### 2.2.2 Algoritmo di Schultze

L'algoritmo di Schultze, conosciuto anche come Schwartz Sequential Dropping (SSD), è una variante di Condorcet basata sulla sommatoria del conteggio delle vittorie tra i concorrenti. Ad esempio, nel caso in cui si hanno 3 candidati, A, B, C, e questi vengono ordinati in una lista di preferenze dalla posizione 1 alla posizione 3 dove la posizione 1 indica la posizione di maggior preferenza, si può avere il seguente risultato:

- Posizione 1: A
- Posizione 2: B
- Posizione 3: C

In questa lista A ottiene 2 punti, B ottiene 1 punto mentre C nessuno. Riprendendo i voti presenti in tabella 2.1 e moltiplicando i punti per il numero di occorrenze, si ottiene la situazione finale mostrata in tabella 2.2

| Concorrente | x 23 | x 19 | x 16 | x 2 | Sommatoria |
|---|---|---|---|---|---|
| A | 2 | 0 | 0 | 1 | 48 |
| B | 0 | 2 | 1 | 0 | 54 |
| C | 1 | 1 | 2 | 2 | 78 |

Tabella 2.2: Tabella delle sommatorie ottenute con l'applicazione dell'algoritmo di Schultze

Prendendo in considerazione le sommatorie in ordine decrescente, otteniamo, il seguente risultato:

- Posizione 1: C
- Posizione 2: B
- Posizione 3: A

Il risultato di Schultze non coincide sempre con il risultato di Condorcet.

## 2.3 Requisiti

La libreria EleKtion fornisce la possibilità di creare ed effettuare votazioni. Essa non tratta la parte preliminare ad una votazione, quale fase di proposta, discussione, eventuali pre-votazioni o quorum, ma pone l'obiettivo sulla definizione e gestione dei concorrenti coinvolti, la tipologia di algoritmo utilizzato per ottenere il risultato finale ed i voti presenti nella fase della votazione. Al termine della stessa, viene resa disponibile una classifica che riporta informazioni circa le specifiche impostate e le caratteristiche dei concorrenti. Ciascuna votazione ha associata una competizione con una metrica che caratterizza l'operato dei concorrenti. Prendendo esempi di riferimento legati al mondo del motor sport, si potrebbe pensare ai punti totalizzati per ogni gara, come nel caso di Formula 1 (F1) o Moto GP. Seppur in molti casi le regole non cambino in modo drastico, ciascuno sport è influenzato da molti fattori: i concorrenti, gli ambienti di gara, i tifosi e altri aspetti che contribuiscono a rendere ogni campionato come un capitolo a sè stante. Inoltre ognuno di essi ha un proprio metodo di calcolo ed organizzazione delle competizioni. Soprattutto in caso di pareggi, è opportuno avere una logica per risolvere il conflitto e determinare un ordine specifico per le classifiche. Se si considerasse l'esito di ciascuna gara come un votazione, cambiando l'algoritmo utilizzato in un campionato, si otterrebbero esiti differenti, o ancora, se si cambiasse la logica di vittoria nelle fasi di un campionato, si potrebbe simulare come sarebbe stato l'esito finale con regole differenti.

### 2.3.1 Requisiti funzionali

- Definizione di gestori delle votazioni, intesi come contenitori delle stesse. Ciascuna votazione deve essere indipendente dalle altre. L'eventuale collegamento tra esse viene demandato all'utilizzatore della libreria;

- Definizione dei concorrenti che rappresentano i candidati disponibili, assieme alle relative caratteristiche ed eventuali punteggi. I punteggi sono collegati ad una specifica metrica di interesse.

- Definizione dell'algoritmo da utilizzare. L'algoritmo deve determinare la logica con cui sono confrontati e selezionati i concorrenti. L'algoritmo può utilizzare la metrica disponibile per effettuare ulteriori valutazioni in occasione di pareggio o altre casistiche definibili dall'utilizzatore. L'algoritmo può essere parametrizzato, per impattare il proprio funzionamento.

- Definizione dei voti presenti nella votazione. La tipologia di voto deve essere dipendente dall'algoritmo utilizzato, e in ogni caso può essere dei seguenti tipi:

  1. Voto a singola preferenza
  2. Voto a lista di preferenze

  Deve essere data la possibilita si anonimizzare il votante, in modo tale che successivamente non sia possibile risalire al suo identificativo;

- Le fasi di definizione devono essere corredate da opportuni controlli di congruità, al fine di elaborare la votazione in modo coerente;

- La classifica finale deve essere disponibile e visualizzabile dall'utilizzatore. Deve esser data evidenza di eventuali situazioni di pareggio, di quale algoritmo è stato utilizzato e delle caratteristiche dei concorrenti;

- Deve essere possibile definire tutti gli elementi necessari attraverso l'uso di costrutti che agevolino la rapidità di creazione, evitando che l'utilizzatore effettui manualmente i controlli minimi ed obbligatori.

### 2.3.2 Requisiti non funzionali

- La libreria deve essere disponibile su molteplici piattaforme, facilmente importabile in esse ed avere documentazione a supporto.

## 2.4 Modello del dominio

EleKtion ha a disposizione uno o più gestori delle votazioni. I gestori sono indipendenti tra loro e così anche le votazioni all'interno degli stessi, fatto salvo l'uso coerente della tipologia della votazione e della metrica di punteggio, che rimane stabile nello stesso gestore. Non è contemplata l'ipotesi di più turni nella stessa votazione, quindi in tal caso devono essere gestiti come votazioni separate. Una votazione ha associata una determinata competizione, alla quale è associato l'elenco dei concorrenti, detti anche candidati. Ciascun concorrente può avere una serie di punteggi, e la tipologia di punteggio è strettamente correlata alla metrica adottata. La votazione, inoltre, ha associato l'elenco dei voti, i quali hanno i riferimenti sia del votato che del votante ma quest'ultimo è rilevante solo in caso di votazione non anonima. È importante che un voto, per essere considerato valido, sia riferito ad un candidato esistente e sia della tipologia ammessa dalla votazione. Infine, ci può essere un solo algoritmo collegato alla votazione, il quale può contenere dei parametri per effettuare variazioni nella configurazione per il suo funzionamento. La votazione ha un'unica classifica finale, la quale può essere consultata. In figura 2.1 viene rappresentato il diagramma UML con le interfacce che modellano il dominio descritto in questo capitolo.

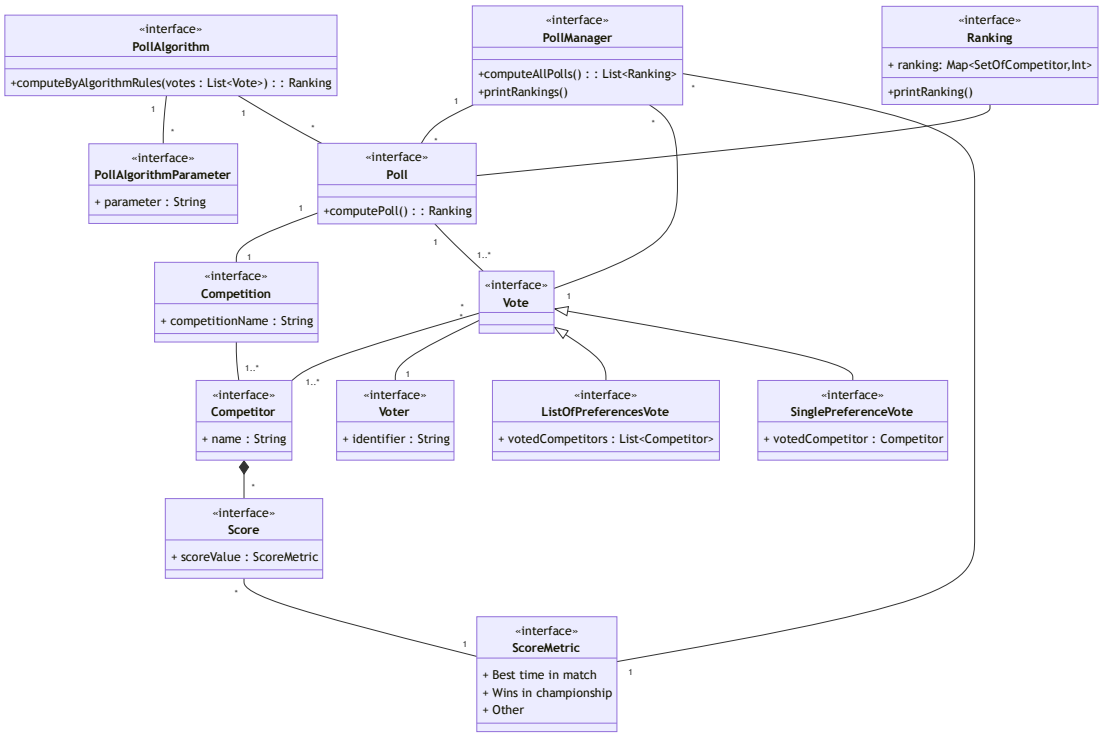


Figura 2.1: Diagramma UML del modello del dominio di EleKtion

# Capitolo 3

# Design

In questo capitolo, partendo dalla modellazione del dominio descritta nel paragrafo 2.4, viene sviluppato un approfondimento sulla definizione delle metriche di punteggio e sulla porzione impattata dall'introduzione dei DSL, volti alla semplificazione dell'utilizzabilità della libreria EleKtion. Ciascuna funzionalità viene approfondita in un apposito paragrafo, ognuno contenente diagrammi UML che esplicano l'organizzazione di interfacce e classi astratte. Questi diagrammi sono realizzati con il presupposto di poter adottare l'utilizzo dei seguenti costrutti:

- Tipi generici[1];
- Overloading degli operatori[2];
- Higher order functions (function literal with receiver)[3];
- Companion object[4].

Si rimanda alle fonti per l'approfondimento specifico in Kotlin, la sintassi utilizzata nei diagrammi è generica ed astrae dal linguaggio specifico.

[1] `https://archive.is/kISda`
[2] `https://archive.is/cExIS`
[3] `https://archive.is/mWF2b`
[4] `https://archive.is/yfHS5`

## 3.1 Domain Specific Language per la definizione del gestore delle votazioni

Questo DSL è pensato per facilitare la creazione del gestore delle votazioni. Il gestore delle votazioni adotta uno specifico tipo `S`, sottotipo di `ScoreMetric`, ed uno specifico tipo di voto `V`, sottotipo di `Vote`. Una metrica è definibile come il criterio tale per cui un concorrente può essere messo a confronto con un altro, sulla base della sua prestazione nella competizione. Pertanto, è rappresentabile come una comparazione tra valori. Dall'interfaccia `ScoreMetric` possono essere definiti numerosi sottotipi utili alla competizione trattata, ad esempio il punteggio di un concorrente può essere legato ai punti assegnati in una gara, alle vittorie ottenute in campionato, oppure a criteri legati al tempo. Per semplificarne la creazione è utile definire una funzione di supporto contenuta in un oggetto `companion`, come mostrato in figura 3.1.

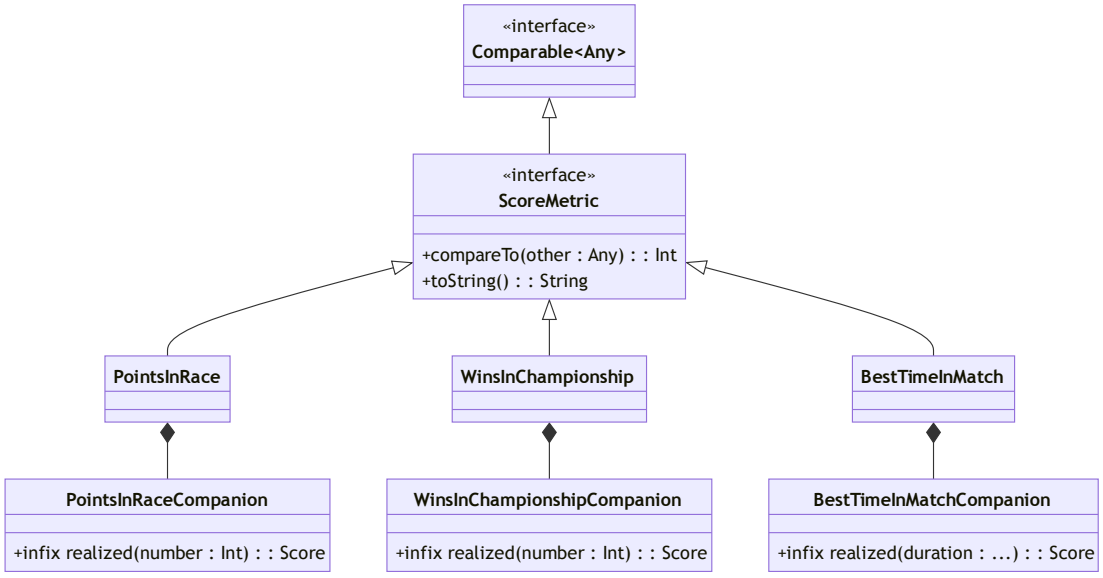


Figura 3.1: Diagramma UML dell'organizzazione di una metrica

I tipi `S` e `V` vincolano i seguenti componenti che verranno trattati, pertanto non è possibile adottare diverse metriche di punteggio o diverse tipologie di voto tra le varie votazioni presenti nel medesimo gestore. Il punto di ingresso dell'intero flusso è rappresentato dalla funzione `initializedAs`, al suo interno è possibile inizializzare una votazione attraverso la funzione `poll`, e l'utilizzo dell'operatore unario `+`, per l'aggiunta di quest'ultima alla lista delle votazioni. In figura 3.2 viene mostrato il diagramma UML delle interfacce e della classe astratta.

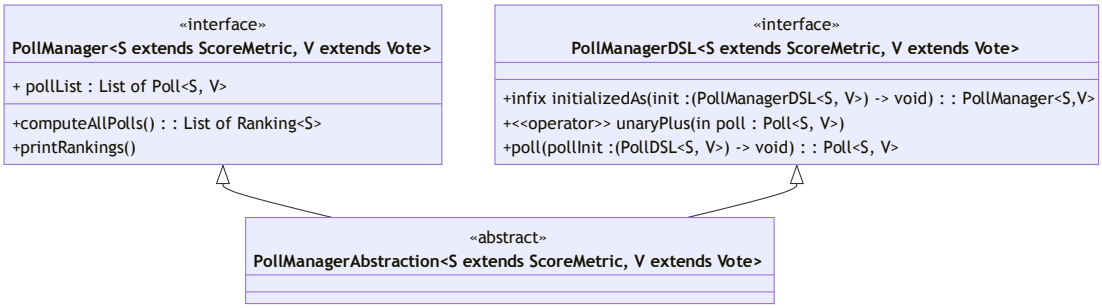


Figura 3.2: Diagramma UML del DSL per la definizione del gestore delle votazioni

## 3.2 Domain Specific Language per la definizione della votazione

Questo DSL è pensato per facilitare la creazione di una votazione. In base al tipo di voto viene definito un costrutto per agevolarne la creazione. Nel caso in cui il voto sia a singola preferenza, è necessario specificare l'identificativo del votato e l'identificativo del votante (vedi funzione `votedBy` in `SinglePreferenceVoteAlgorithmDSL`, figura 3.3). Se quest'ultimo non deve essere pubblico, corrisponderà ad un identificativo casuale (vedi funzione `asAnonymousVote` in `SinglePreferenceVoteAlgorithmDSL`, figura 3.3). Nel caso in cui il voto sia a lista di preferenze, per costruire la lista è necessario specificare l'elenco degli identificativi dei votati, utilizzando la funzione `then` in concatenazione (vedi funzione `then` in `ListOfPreferencesVoteAlgorithmDSL`, figura 3.4). Allo stesso modo del voto a singola preferenza, è prevista la possibilità di anonimizzare il votante. Sono previsti metodi per semplificare l'inizializzazione degli algoritmi disponibili:

- `majorityVotesAlgorithm`: rappresenta la creazione di un algoritmo a maggioranza, in cui il vincitore è colui che ricevuto più voti;
- `majorityVotesHScoreAlgorithm`: rappresenta la creazione di un algoritmo a maggioranza, in cui il vincitore è colui che ricevuto più voti. Se si verificano pareggi, viene selezionato il concorrente che ha il punteggio maggiore, secondo la metrica di punteggio definita;
- `majorityVotesLScoreAlgorithm`: rappresenta la creazione di un algoritmo a maggioranza, in cui il vincitore è colui che ricevuto più voti. Se si verificano pareggi, viene selezionato il concorrente che ha il punteggio minore, secondo la metrica di punteggio definita;
- `condorcetAlgorithm`: rappresenta la creazione dell'algoritmo di Condorcet;
- `schultzeAlgorithm`: rappresenta la creazione dell'algoritmo di Schultze.

I parametri utili al funzionamento dell'algoritmo possono essere impostati utilizzando l'operatore unario `+` presente in `PollAlgorithmDSL`. In figura 3.5 ven-

gono mostrati gli altri metodi utili alla creazione di una votazione: la funzione `competition` istanzia una competizione, richiedendo che vengano definiti il nome della competizione ed un inizializzatore in cui saranno specificati i candidati presenti. La competizione, così come l'algoritmo scelto, sono effettivamente assegnati alla votazione attraverso l'uso dell'operatore unario `-`. Infine è possibile aggiungere un elenco di voti, istanziati utilizzando i metodi descritti nella prima parte di questo paragrafo. Ciascun voto viene effettivamente aggiunto alla lista di voti attraverso l'uso dell'operatore unario `+`. Per motivi di spazio, nelle figure 3.3 e 3.4 sono stati utilizzati gli acronimi `SP` e `LOP`, al posto di, rispettivamente, `SinglePreference` e `ListOfPreferences`.

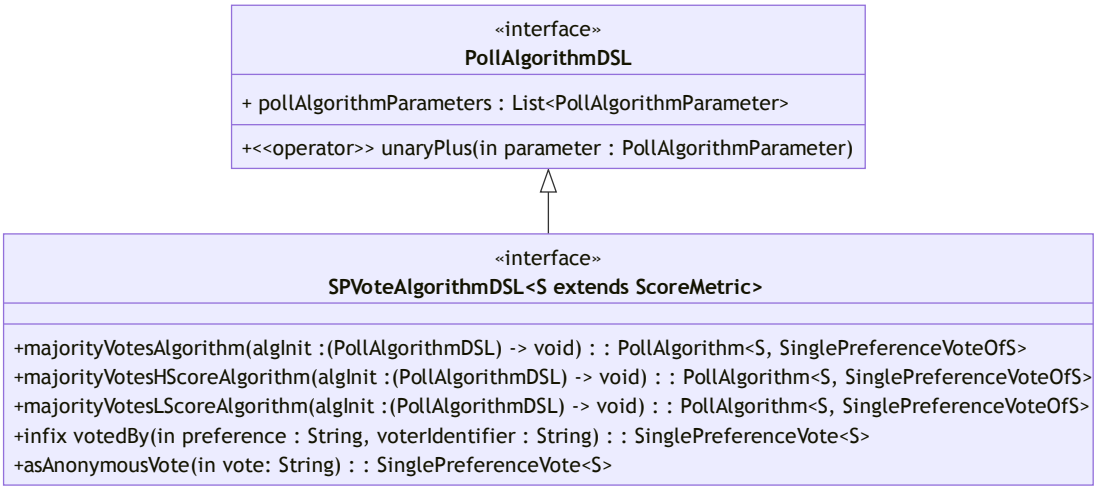


Figura 3.3: Diagramma UML del DSL per la definizione dell'algoritmo di votazione a singola preferenza

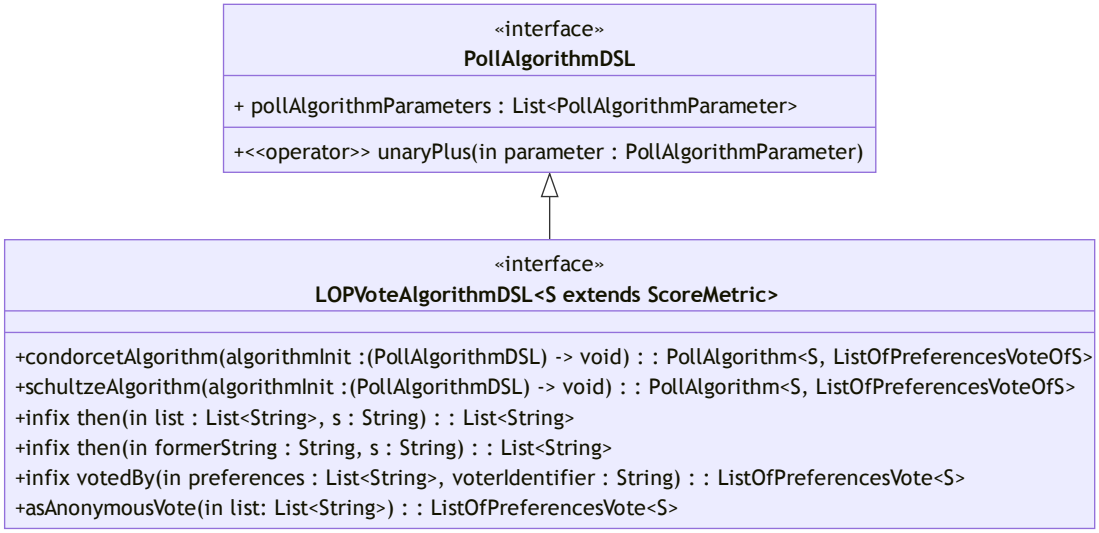


Figura 3.4: Diagramma UML del DSL per la definizione dell'algoritmo di votazione a lista di preferenze

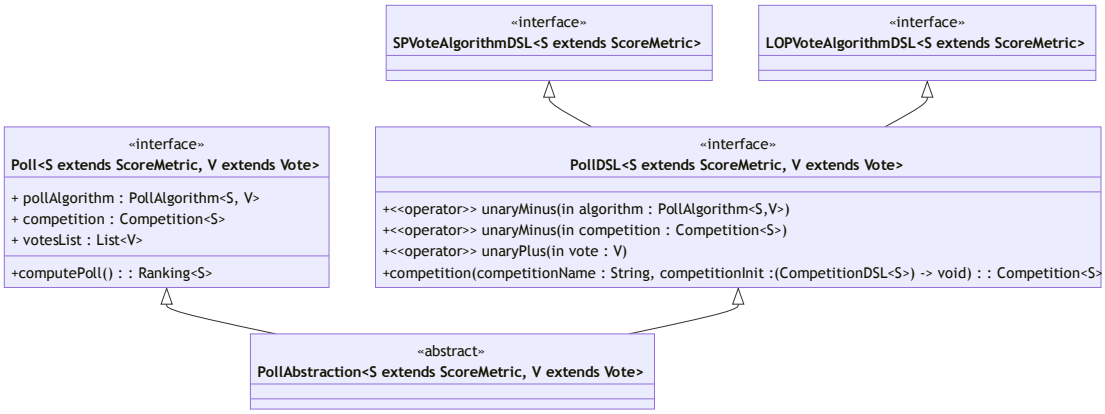


Figura 3.5: Diagramma UML del DSL per la definizione della votazione

## 3.3 Domain Specific Language per la definizione della competizione

Questo DSL è pensato per facilitare la creazione ed il popolamento di una competizione. La funzione `competitor` istanzia un concorrente, richiedendo che sia specificato il proprio identificativo ed un inizializzatore contenente i punteggi realizzati. Il concorrente istanziato, viene aggiunto alla lista dei concorrenti grazie all'uso dell'operatore unario `+` . In figura 3.6 viene mostrato il diagramma UML delle interfacce e della classe astratta.

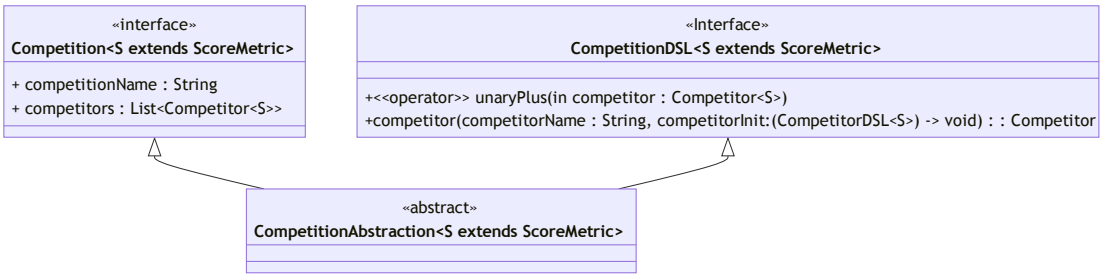


Figura 3.6: Diagramma UML del DSL per la definizione della competizione

## 3.4 Domain Specific Language per la definizione del concorrente

Questo DSL è ideato per facilitare la creazione di un concorrente. Ciascun concorrente può avere un elenco di punteggi, i quali vengono assegnati attraverso l'uso dell'operatore unario +.

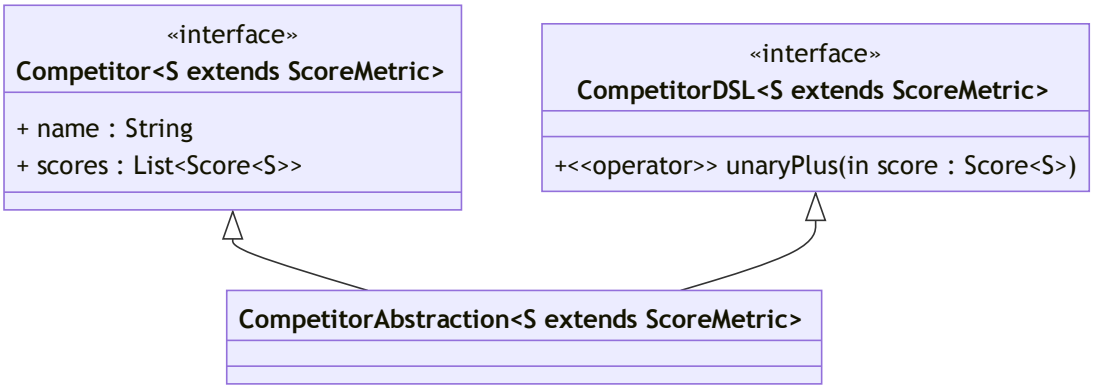


Figura 3.7: Diagramma UML del DSL per la definizione del concorrente

# Capitolo 4

# Implementazione

In questo capitolo vengono mostrati alcuni casi d'uso realizzati per verificare il corretto funzionamento della libreria, dando nota delle possibili eccezioni che sono state gestite. Successivamente, viene fornita una panoramica sul processo di *build automation* e vengono mostrati gli step realizzati per volgere alla creazione e pubblicazione della libreria. Per la libreria, il cui codice sorgente è reperibile all'indirizzo https://github.com/corinz97/EleKtion,
sono stati adottati Kotlin come linguaggio di programmazione e Gradle come tool a supporto della *build automation*.

## 4.1 Casi realizzati per i differenti tipi di algoritmo

Nel paragrafo 2.4 e nel capitolo 3 sono stati presentati l'organizzazione del dominio da implementare, assieme ad un approfondimento sulle entità che compongono il linguaggio DSL ideato appositamente. In questo paragrafo viene spiegato come vengono gestiti eventuali pareggi ed inconsistenze.

### 4.1.1 Votazioni a singola preferenza

Per gestire le votazioni a singola preferenza è stato creato inizialmente il caso base per la votazione a maggioranza, nella quale il vincitore è il concorrente che riceve più voti, a prescindere dalla presenza di altri criteri come i punteggi. In alcuni casi è possibile avere parità tra i voti, risultando più vincitori nella stessa posizione della classifica. Successivamente è stata estesa la gestione dei pareggi, dando la possibilità all'utilizzatore della libreria di definire una metrica per la competizione ed un elenco di punteggi realizzati da ciascun concorrente, basati su quella metrica. Nel caso in cui si ha un pareggio di voti è possibile decidere di effettuare un ulteriore sotto-confronto basato sui punteggi, per selezionare chi risulta vincitore sulla base del punteggio più basso o più alto realizzato nella competizione. Questo sotto-confronto può portare ad avere dei vincitori a pari merito, oppure di piazzarli in posizioni differenti, proprio per la differenza dei punteggi. La gestione quindi non garantisce l'assenza totale di pareggi, che possono verificarsi anche a livello di punteggio. Inoltre, è possibile avere un parametro che indica la capacità di effettuare molteplici voti per votante, ma di negargli comunque l'opportunità di votare più volte lo stesso candidato. Le classi realizzate per gestire tutte queste casistiche sono `MajorityVotesAlgorithm`, `MajorityVotesThenHighestScoreAlgorithm` e `MajorityVotesThenLowestScoreAlgorithm`. Nell'estratto di codice 4.1 vengono mostrati i controlli effettuati prima di compiere la scelta del vincitore.

Estratto di codice 4.1: Controlli effettuati prima di compiere la scelta del vincitore, nella votazione a singola preferenza

```
1  if (candidates.groupingBy { it.name }.eachCount().any { it.value > 1 }) {
2          error("Candidate already declared")
3      }
4      require(votes.any())
5      if (votes.map { it.votedCompetitor }.any { it !in candidates }) {
6          error("Voted candidate doesn't exist as object")
7      }
8      when (pollAlgorithmParameters.count { it == ConstantParameter.
           AllowMultipleVoteInPollParameter }) {
9          0 -> {
10             if (votes.groupingBy { it.voter.identifier }.eachCount().any { it.
                   value > 1 }) {
11                 error("Each voter can vote only once")
```

```
            }
        }
        1 -> {
            if (votes.groupingBy {
                    Pair(it.votedCompetitor.name, it.voter.identifier)
                }.eachCount().any { it.value > 1 }
            ) {
                error("Each voter can vote just once for each competitor")
            }
        }
        else -> error("Parameter can't be repeated more than once")
    }
```

La classifica generata contiene l'insieme dei vincitori con il relativo numero di voti e per ciascuno viene riportato il punteggio maggiore o quello minore. Nel caso in cui si utilizzi l'algoritmo a maggioranza senza sotto-confronto, vengono riportati tutti i punteggi realizzati, inoltre per quest'ultimo non è obbligatorio associare dei punteggi ai giocatori. Negli estratti di codice 4.2, 4.3 e 4.4 vengono mostrati alcuni esempi realizzati: nella definizione di ogni competizione è essenziale dichiarare l'elenco di tutti i possibili candidati in modo da poter verificare che non ci siano voti associati ad un candidato non esistente. Questo controllo è effettuato anche dal DSL nella fase di definizione dei voti, e quindi in precedenza all'invocazione dell'algoritmo, il quale replica questo controllo in quanto è possibile utilizzarlo indipendentemente dall'adozione di DSL. Si definiscono poi l'algoritmo da utilizzare e l'elenco di voti. I voti possono essere anonimizzati, generando una stringa *UUID* che viene usata come identificativo del votante.

Estratto di codice 4.2: Votazione a singola preferenza, in cui il vincitore è competitor1.

```
val election =
    PollManagerInstance<BestTimeInMatch, SinglePreferenceVote<BestTimeInMatch>>()
        initializedAs {
        +poll {
                -competition("Race 1") {
                    +competitor("competitor1") {}
                    +competitor("competitor2") {}
                }
                -majorityVotesAlgorithm {
                    +ConstantParameter.AllowMultipleVoteInPollParameter
                }
```

```
                +("competitor1".asAnonymousVote())
                +("competitor2".asAnonymousVote())
                +("competitor1".asAnonymousVote())
            }
    }
election.printRankings()
```

Estratto di codice 4.3: Votazione a singola preferenza, in cui il vincitore è competitor1.

```
val election =
    PollManagerInstance<BestTimeInMatch, SinglePreferenceVote<BestTimeInMatch>>()
        initializedAs {
        +poll {
                -competition("Race 2") {
                    +competitor("competitor1") {
                        +(BestTimeInMatch realized (0.toDuration(DurationUnit.
                            HOURS)))
                        +(BestTimeInMatch realized (1.toDuration(DurationUnit.
                            HOURS)))
                        +(BestTimeInMatch realized (2.toDuration(DurationUnit.
                            HOURS)))
                    }
                    +competitor("competitor2") {
                        +(BestTimeInMatch realized (1.toDuration(DurationUnit.
                            HOURS)))
                    }
                }
                -majorityVotesHScoreAlgorithm {
                    +ConstantParameter.AllowMultipleVoteInPollParameter
                }
                +("competitor2" votedBy "voter1")
                +("competitor1" votedBy "voter1")
                +("competitor2" votedBy "voter2")
                +("competitor1" votedBy "voter2")
        }
    }
election.printRankings()
```

Estratto di codice 4.4: Votazione a singola preferenza, in cui competitor1 e competitor2 pareggiano.

```
val election =
    PollManagerInstance<BestTimeInMatch, SinglePreferenceVote<BestTimeInMatch>>()
        initializedAs {
        +poll {
```

```
4                 -competition("Race 3") {
5                     +competitor("competitor1") {
6                         +(BestTimeInMatch realized (0.toDuration(DurationUnit.
                              HOURS)))
7                         +(BestTimeInMatch realized (1.toDuration(DurationUnit.
                              HOURS)))
8                         +(BestTimeInMatch realized (2.toDuration(DurationUnit.
                              HOURS)))
9                     }
10                    +competitor("competitor2") {
11                        +(BestTimeInMatch realized (0.toDuration(DurationUnit.
                              HOURS)))
12                    }
13                }
14                -majorityVotesLScoreAlgorithm {
15                    +ConstantParameter.AllowMultipleVoteInPollParameter
16                }
17                +("competitor2" votedBy "voter1")
18                +("competitor1" votedBy "voter1")
19                +("competitor2" votedBy "voter2")
20                +("competitor1" votedBy "voter2")
21        }
22    }
23 election.printRankings()
```

Sono stati realizzati test a supporto dell'utilizzo del DSL e degli algoritmi, in particolare è stato verificato che venga restituita un'eccezione nei casi in cui:

1. Nell'ambito degli algoritmi, in una determinata votazione

   - un candidato sia presente più di una volta nell'insieme dei candidati;
   - non siano presenti voti;
   - un votante effettui più di un voto, a meno di passare all'algoritmo l'apposito parametro;
   - il parametro, quando presente, sia dichiarato più di una volta;
   - un votante voti due volte lo stesso candidato, seppur in presenza del parametro;
   - un votante voti un candidato non esistente;
   - la classifica generata non contenga alcun elemento;

- la classifica generata restituisca i risultati ordinati in modo non decrescente di voti;
- gli algoritmi che utilizzano il sotto-criterio rilevino che ci sono concorrenti senza alcun punteggio.

2. Nell'ambito del DSL

- non siano presenti voti;
- si dichiari lo stesso candidato più di una volta nell'insieme di candidati;
- si dichiari un voto per un candidato non esistente

### 4.1.2 Votazioni a lista di preferenze

Per gestire la votazioni a lista di preferenze è stato necessario procedere all'implementazione dell'algoritmo di Condorcet e dell'algoritmo di Schultze. Come è stato descritto nel paragrafo 2.2, quest'ultimo è una variante del primo, pertanto condividono buona parte della logica. Inizialmente usando una matrice di dimensione $n$ $x$ $n$, dove $n$ è il numero dei candidati, ed esaminando l'indice in cui si posizionano i candidati in un voto, si evidenzia chi dei due abbiamo ottenuto un punto. Si ottiene quindi una matrice che indica quante volte sono stati vinti gli scontri di coppia tra due determinati candidati. Successivamente, l'algoritmo di Condorcet effettua tante iterazioni quanti sono i concorrenti rimasti da piazzare in graduatoria, ed in ogni iterazione si esamina quali siano i candidati che hanno battuto tutti gli altri negli scontri a coppie. Anche l'algoritmo di Schultze effettua tante iterazioni quanti sono i concorrenti rimasti da piazzare in graduatoria, ed in ogni iterazione si determinano quali sono i candidati che hanno ottenuto il maggior numero di vittorie negli scontri a coppie. Pertanto, è stata ammessa la possibilità di ottenere dei pareggi a parità di posizionamento. Anche per questa tipologia di votazione è possibile avere un parametro che indica la capacità di effettuare molteplici voti per votante , ma di negargli comunque l'opportunità di votare più volte la stessa lista. Le classi realizzate per gestire le casistiche sono

`CondorcetAlgorithm` e `SchultzeAlgorithm`. Nell'estratto di codice 4.5 vengono mostrati i controlli effettuati prima di eseguire l'algoritmo di votazione.

Estratto di codice 4.5: Controlli effettuati prima di eseguire l'algoritmo di votazione a lista di preferenze

```
if (candidates.groupingBy { it.name }.eachCount().any { it.value > 1 }) {
        error("Candidate already declared")
    }
    require(votes.any())
    when (pollAlgorithmParameters.count { it == ConstantParameter.
        AllowMultipleVoteInPollParameter }) {
        0 -> {
            if (votes.groupingBy { it.voter.identifier }.eachCount().any { it.
                value > 1 }) {
                error("Each voter can vote only once")
            }
        }
        1 -> {
            if (votes.groupingBy {
                    Pair(it.votedCompetitors.map { c -> c.name }, it.voter.
                        identifier)
                }.eachCount().any { it.value > 1 }
            ) {
                error("Each voter can vote just once for each list of preferences"
                    )
            }
        }
        else -> error("Parameter can't be repeated more than once")
    }
    votes.map { it.votedCompetitors }.forEach {
        val setOfCompetitors = it.toSet()
        if (setOfCompetitors != candidates) {
            if ((setOfCompetitors - candidates).isNotEmpty()) {
                error("A list of preferences contains one o more not allowed
                    candidate")
            }
            if ((candidates - setOfCompetitors).isNotEmpty()) {
                error("Every allowed candidate must be present in every list of
                    preferences")
            }
        }
        if (it.groupingBy { comp -> comp }.eachCount().any { count -> count.value
            > 1 })
            error("Every allowed candidate can be present only once in the list of
                 competitors")
    }
```

Infine viene generata la classifica contenente l'elenco dei vincitori, posti in ordine di preferenza decrescente secondo la logica dell'algoritmo selezionato. A parità di posizione è possibile trovare più di un candidato e non è previsto riportare il numero di voti, poichè è scollegato dalla logica algoritmica. Negli estratti di codice 4.6 e 4.7 vengono mostrati alcuni esempi realizzati: nella definizione di ogni competizione è essenziale dichiarare l'elenco di tutti i possibili candidati in modo da poter verificare che non ci siano voti associati ad un candidato non esistente. Questo controllo è effettuato anche dal DSL nella fase di definizione dei voti, e quindi in precedenza all'invocazione dell'algoritmo, che replica questo controllo in quanto è possibile utilizzarlo indipendentemente dall'adozione di DSL. Si definiscono poi l'algoritmo da utilizzare e l'elenco di voti. I voti possono essere anonimizzati, come nel caso delle votazioni a preferenza singola.

Estratto di codice 4.6: Votazione a lista di preferenze con l'algoritmo di Condorcet, in cui la classifica generata è competitorC, competitorA, competitorB

```
1  var counter = 1
2  val election =
3      PollManagerInstance<BestTimeInMatch, ListOfPreferencesVote<BestTimeInMatch>>()
           initializedAs {
4          +poll {
5                  -competition("Sport match") {
6                      +competitor("competitorB") {}
7                      +competitor("competitorA") {}
8                      +competitor("competitorC") {}
9                  }
10                 -condorcetAlgorithm {}
11                 +("competitorA" then "competitorB" then "competitorC" votedBy "
                       anonym" + counter++)
12                 +("competitorA" then "competitorB" then "competitorC" votedBy "
                       anonym" + counter++)
13                 +("competitorC" then "competitorA" then "competitorB" votedBy "
                       anonym" + counter++)
14                 +("competitorC" then "competitorA" then "competitorB" votedBy "
                       anonym" + counter++)
15                 +("competitorC" then "competitorA" then "competitorB" votedBy "
                       anonym" + counter++)
16                 +("competitorC" then "competitorA" then "competitorB" votedBy "
                       anonym" + counter++)
17                 +("competitorB" then "competitorC" then "competitorA" votedBy "
                       anonym" + counter++)
```

```
18                 +("competitorA" then "competitorC" then "competitorB" votedBy "
                       anonym" + counter++)
19                 +("competitorB" then "competitorA" then "competitorC" votedBy "
                       anonym" + counter++)
20                 +("competitorC" then "competitorB" then "competitorA" votedBy "
                       anonym" + counter++)
21         }
22     }
23 election.printRankings()
```

Estratto di codice 4.7: Votazione a lista di preferenze con l'algoritmo di Schultze, in cui la classifica generata è competitorC, competitorA, competitorB

```
1  val election =
2      PollManagerInstance<BestTimeInMatch, ListOfPreferencesVote<BestTimeInMatch>>()
           initializedAs {
3          +poll {
4                  -competition("Sport match") {
5                      +competitor("competitorB") {}
6                      +competitor("competitorA") {}
7                      +competitor("competitorC") {}
8                  }
9                  -schultzeAlgorithm {}
10                 +(("competitorA" then "competitorB" then "competitorC").
                       asAnonymousVote())
11                 +(("competitorA" then "competitorB" then "competitorC").
                       asAnonymousVote())
12                 +(("competitorC" then "competitorA" then "competitorB").
                       asAnonymousVote())
13                 +(("competitorC" then "competitorA" then "competitorB").
                       asAnonymousVote())
14                 +(("competitorC" then "competitorA" then "competitorB").
                       asAnonymousVote())
15                 +(("competitorC" then "competitorA" then "competitorB").
                       asAnonymousVote())
16                 +(("competitorB" then "competitorC" then "competitorA").
                       asAnonymousVote())
17                 +(("competitorA" then "competitorC" then "competitorB").
                       asAnonymousVote())
18                 +(("competitorB" then "competitorA" then "competitorC").
                       asAnonymousVote())
19                 +(("competitorC" then "competitorB" then "competitorA").
                       asAnonymousVote())
20         }
21     }
22 election.printRankings()
```

Sono stati realizzati test a supporto dell'utilizzo del DSL e degli algoritmi, in particolare è stato verificato che venga restituita un'eccezione nei casi in cui:

1. Nell'ambito degli algoritmi, in una determinata votazione
    - un candidato sia presente più di una volta nell'insieme dei candidati;
    - un votante effettui più di un voto, a meno di passare all'algoritmo l'apposito parametro;
    - il parametro, quando presente, sia dichiarato più di una volta;
    - un votante voti due volte la stessa lista, seppur in presenza del parametro;
    - un votante voti una lista in cui appaia un candidato non esistente;
    - un votante voti una lista in cui sia stato omesso almeno un candidato;
    - un votante voti una lista in cui un candidato appaia più volte;
    - non siano presenti voti;
    - la classifica generata non contenga alcun elemento.
2. Nell'ambito del DSL
    - si dichiari un candidato più di una volta nell'insieme di candidati;
    - si dichiari un voto per un candidato non esistente;
    - si dichiari un voto in cui un candidato appaia più volte;
    - non siano presenti voti.

Utilizzando il DSL è possibile dichiarare voti in cui non appaiono tutti i possibili candidati e in tal caso, i mancanti saranno piazzati in fondo alla lista di preferenze, ossia come più sfavoriti.

## 4.2 Build automation

L'adozione di un sistema di *build automation* è un fattore chiave per la produzione efficace ed efficiente di software di qualità. Come principio base, è importante che

ciascuna modifica apportata al codice sia testata a livello unitario e abbia documentazione correlata, che sia la pubblicazione di una nuova funzionalità piuttosto che la variazione di un campo, a titolo meramente esemplificativo. Inoltre, queste modifiche devono essere ulteriormente verificate attraverso test di integrazione, in modo da garantire la piena compatibilità con i componenti pre-esistenti. Le verifiche possono essere effettuate manualmente dallo sviluppatore, ma per ridurre il rischio di errori è possibile adottare un sistema di automazione, che attraverso una serie di task predefiniti e ripetibili, verifichi automaticamente gli step, gestendo anche le dipendenze tra gli stessi. Così facendo, si può subito notare se tutto è andato a buon fine oppure se è necessario effettuare interventi correttivi. Una volta che il flusso di compilazione e verifica viene terminato con successo, è possibile procedere alla generazione di uno o più artefatti e della relativa documentazione. Questi prodotti possono essere sottoposti ad un'operazione di versionamento, in modo da distinguere le evoluzioni del software e semplificarne la distribuzione e la tracciabilità. Poichè gli artefatti generati possono essere dipendenti dall'architettura della macchina e dal sistema operativo sui quali avviene il processo di *build*, è opportuno eseguirlo su piattorme dedicate come GitHub (GH). GH fornisce ambienti standard e personalizzabili in base a direttive, gestendo il processo di Continuous Integration (CI)/Continuous Delivery (CD) che è stato descritto in questo paragrafo. Infine, per favorire la distribuzione e la compatibilità verso molteplici sistemi, è utile adottare strumenti come Kotlin Multiplatform, che semplificano la generazione dell'output finale adattandolo alle specifiche piattaforme.

### 4.2.1 Kotlin Multiplatform

Kotlin Multiplatform è un tool che favorisce il riuso di codice tra molteplici piattaforme. Grazie ad esso, riutilizzando la logica applicativa scritta in Kotlin, è possibile produrre artefatti compatibili in molteplici piattaforme (dette *target*), come Java Virtual Machine (JVM), Javascript (JS), Android, iOS, oltre a piattaforme native come Linux, Windows, macOS. I *target* sono disposti secondo una gerarchia predefinita, comunque modificabile in caso di necessità, e sono previsti *target* intermedi. Per ogni *target* viene definito un *source set*, ossia un insieme

di file sorgente, che definisce anche le dipendenze e le compatibilità. Il *source set* predefinito è `commonMain` e contiene il codice comune a tutti i *target*. Al suo interno è possibile utilizzare funzioni e dipendenze che siano compilabili verso tutte le piattaforme dichiarate, mentre quelle che sono legate ad uno specifico linguaggio o architettura vanno definite ed utilizzate all'interno degli opportuni *source sets*, come `jvmMain` e `jsMain`. Un esempio è visibile in figura 4.1, in cui i *target* intermedi sono colorati in verde: *apple* conterrà codice compatibile solamente con i *target* che lo estendono, oltre a quello ereditato dai livelli superiori. Pertanto, è ammesso che *macos* utilizzi funzioni che non siano disponibili in *tvos* ma entrambi accedono al codice messo a disposizione da *apple*. Inoltre, questo meccanismo permette di applicare un *template method* architetturale, grazie al quale è possibile dichiarare nel *source set* `commonMain` funzioni agnostiche tramite la direttiva `expected`, senza definirne il contenuto, che sarà valorizzato in un successivo momento. In fase di compilazione, nelle versioni platform-specific, saranno valorizzate tramite funzioni che adottano la direttiva `actual`, che posso richiamare funzionalità altrimenti non disponibili in `commonMain`. Una logica analoga è applicabile anche alla sezione dei test: sono disponibili un *source set* di base, detto `commonTest`, e altri eventuali che contengono codice compatibile solo nella relativa piattaforma. In figura 4.2 viene mostrato il flusso e gli output ottenuti utilizzando i *target JVM* e *JS*.

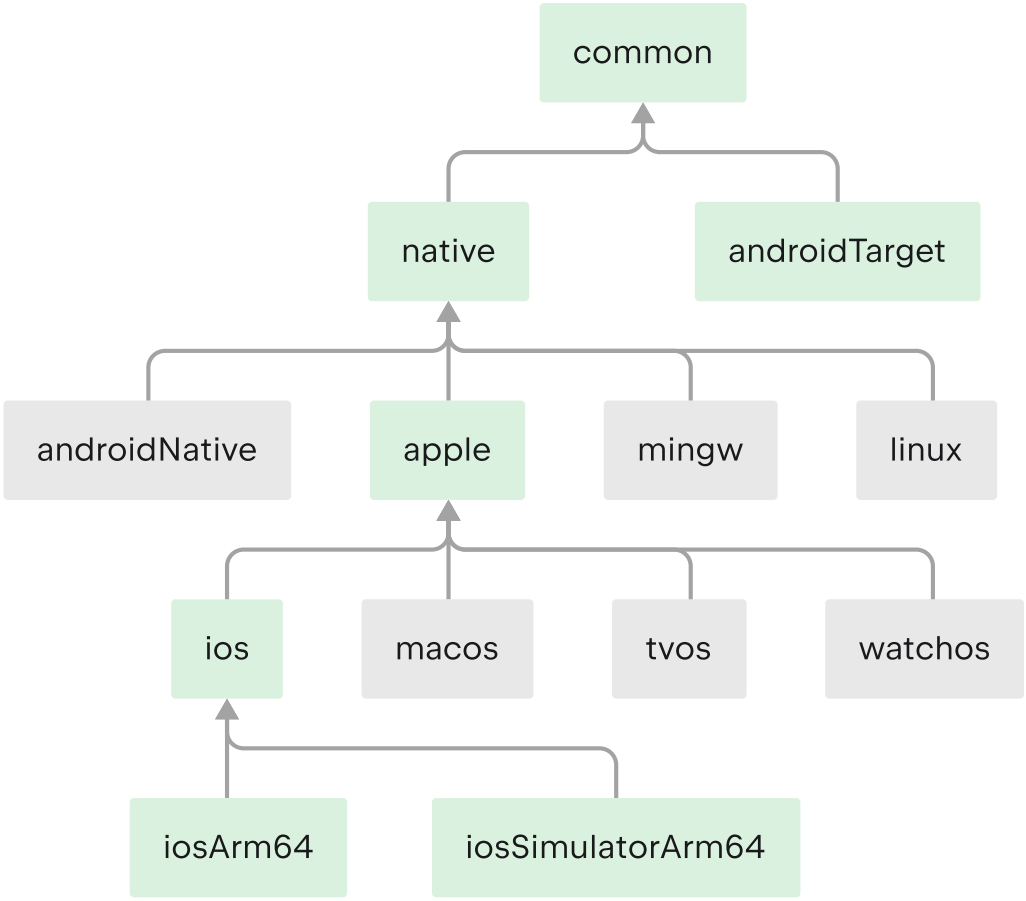


Figura 4.1: Esempio di una possibile gerarchia di *target*. Fonte: `https://archive.is/eH6ha`

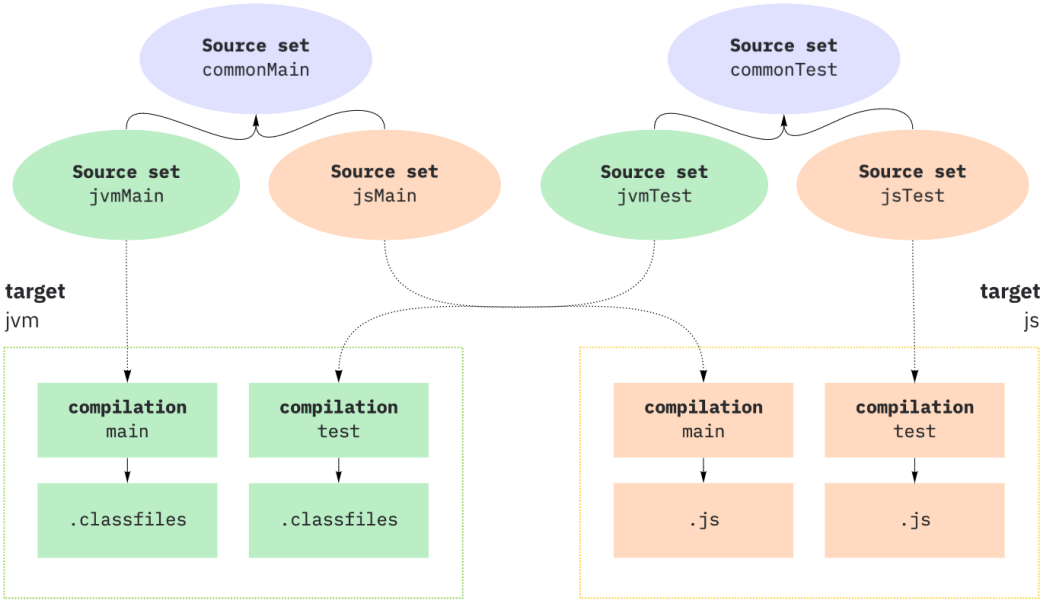


Figura 4.2: Esempio di output ottenibili in un progetto multipiattaforma. Fonte: `https://archive.is/wSykW`

### 4.2.2 GitHub Actions

GitHub Actions (GH Actions) è una piattaforma di CI/CD basata su repository GH, ai quali possono essere agganciati degli elementi detti *workflow*. Un *workflow*, definibile attraverso la sintassi YAML, rappresenta una sequenza di azioni che possono essere eseguite all'occorrenza di un evento, ad esempio la creazione di un *tag* nel repository, una *richiesta pull*, un'esecuzione manuale richiesta dal proprietario del repository, ecc... In ogni *workflow* va specificato il nome dello stesso e l'insieme di eventi che ne scatena l'esecuzione. Di seguito a ciò, è possibile definire un insieme di *job*. Ciascun *job* è legato ad uno specifico ambiente di esecuzione: va definito che tipo di sistema operativo deve essere adottato (Linux, Windows, macOS) e se al termine dello stesso debba essere restituito un output, che sarà accessibile da altri *job*. È possibile specificare delle variabili in una *matrice*, in modo eseguire parallelamente molteplici istanze di uno stesso *job* in base alle combinazioni delle

variabili; a titolo esemplificativo, eseguire la compilazione di codice utilizzando la stessa versione del compilatore ma su tre sistemi operativi differenti. Ciascun *job* è caratterizzato da un proprio flusso indipendente, di conseguenza per orchestare i processi è opportuno specificare condizioni basate sulle variabili di output oppure sull'indicazione del completamento di uno o più *jobs* desiderati. All'interno del *job* sono definibili uno o più *step*: questi vengono eseguiti in maniera sequenziale e sono impostabili per eseguire una sequenza arbitraria di comandi bash oppure per eseguire una *action* pubblica. Quest'ultima rappresenta un ottimo modo di favorire il riuso e la condivisione di codice nella community GH, in quanto è possibile, ad esempio, utilizzare un *workflow* complesso come se fosse un singolo step (*composite action*) oppure lanciare l'esecuzione di un container senza imporre *effort* riguardo alla gestione delle dipendenze e delle configurazioni (*docker container action*).

### 4.2.3 GitHub Pages

GH Pages è un servizio di hosting disponibile per ogni account ed organizzazione in GH, attraverso il quale è possibile pubblicare un sito web statico e renderlo accessibile tramite un webserver. La pubblicazione avviene attraverso un apposito *workflow* predefinito. Nel caso di un'iniziativa progettuale come una libreria, può essere utilizzato per dare informazioni su come scaricare ed installare il software oppure per pubblicare la documentazione.

### 4.2.4 Maven Central, GitHub Packages e NPM

Maven Central è tra i gestori di repository più popolari per distribuire e riutilizzare artefatti basati su JVM. Un repository è definibile come "la directory che memorizza tutti i progetti e le librerie JAR, i plugin o qualsiasi altro artefatto specifico del progetto che può essere facilmente utilizzato da Maven" [KB22]. Il sistema di gestione delle dipendenze di Maven è in grado di risolvere automaticamente le dipendenze transitive, semplificando la gestione manuale delle dipendenze per gli sviluppatori. Gli artefatti caricati su Maven Central, tra cui librerie di codice compilato ed elenchi di dipendenze, sono immutabili e non possono essere modifi-

cati né eliminati. Gli artefatti vengono identificati univocamente con una tripletta *groupId:artifactId:version*, dove *groupId* identifica un gruppo di artefatti, *artifactId* si riferisce al nome della libreria e *version* identifica univocamente ogni rilascio della libreria. Ad esempio, la tripletta 'org.jetbrains.kotlinx:kotlinx-serialization-json:1.6.0' identifica la versione 1.6.0 di un serializzatore json per Kotlin, gestito da JetBrains. Tutte le versioni di una libreria rilasciate su Maven Central sono sempre disponibili nel repository. Gli utenti della libreria sono liberi di decidere quale versione utilizzare e per quanto tempo e a questi ultimi è delegata la responsabilità di aggiornare le proprie dipendenze per adeguarsi a problemi di sicurezza o evoluzioni dell'API. Nel lungo termine, queste decisioni determinano le librerie e le versioni più popolari nell'intero ecosistema software. Si osserva che per la maggior parte delle librerie, più di una versione è attivamente utilizzata in un dato momento [SVBH+19]. Dallo studio empirico condotto in [SVBH+19], è emerso che "circa il 40% delle librerie ha due o più versioni attivamente utilizzate, mentre quasi il 4% non ha mai avuto alcun utente su Maven Central. Inoltre, si è scoperto che più del 90% delle versioni più popolari non sono le ultime versioni rilasciate, e sia le versioni attive che quelle significativamente popolari sono distribuite lungo la storia delle versioni della libreria".

Un'alternativa all'uso di Maven Central consiste in GitHub Packages (GH Packages). Quest'ultimo pone una gestione semplificata per la pubblicazione degli artefatti, in quanto strettamente connesso all'uso di repository GH, il che permette di avere un controllo degli accessi efficace come quello disponibile per i repo. Al pari di Maven Central viene mantenuto il concetto di artefatti immutabili, e in più viene dato il supporto per l'archiviazione di pacchetti in numerosi linguaggi, tra cui `C#`.

Un altro famoso gestore di pacchetti è Node Package Manager (NPM), focalizzato sulla diffusione di librerie JS e Node.js, si differenzia dai precedenti in quanto non adotta il principio di immutabilità. In questo caso, la modifica o eliminazione di una versione di un pacchetto può avere impatti[1] notevoli sugli utilizzatori, portando a potenziali disastri mondiali.

[1] `https://archive.is/qMtZE`

## 4.3 Generazione multipiattaforma degli artefatti e pubblicazione

La libreria EleKtion è stata realizzata partendo da un template[2] disponibile su GH, che fornisce la struttura di avvio per un progetto multipiattaforma in Kotlin, volto alla generazione e pubblicazione di artefatti. I *target* sono tutti compilati per JVM versione 1.8, successivamente vengono anche convertiti in JS per la generazione della libreria in ambito Web, e viene generata la versione native, compatibile in linguaggio nativo con i seguenti sistemi:

- Linux x64;
- linux arm64;
- mingw x64;
- macos x64;
- macos arm64;
- ios arm64;
- ios x64;
- ios simulator arm64;
- watchos arm32;
- watchos x64;
- watchos simulator arm64;
- tvos arm64;
- tvos x64;
- tvos simulator arm64.

[2] `https://archive.is/rLXB0`

Il processo di automazione viene eseguito grazie ai *workflow* presenti nel repository GH: il primo *workflow*, definito `dispatcher`, viene avviato al momento della ricezione di un comando *push* o della creazione di una *richiesta pull*, e richiama i seguenti *workflow*:

1. `build-and-deploy`: compila i sorgenti ed esegue i test nelle piattaforme dichiarate, effettua controlli di correttezza del codice ed infine pubblica gli artefatti generati;

2. `publish-docs`: genera la documentazione degli artefatti e la mette a disposizione per la pubblicazione.

I *workflow* `dispatcher` e `build-and-deploy` sono forniti dal template ma sono stati opportunamente configurati per lo scopo del progetto, mentre il *workflow* `publish-docs` è stato creato ex-novo.

### 4.3.1 Generazione e pubblicazione della libreria

Il *workflow* `build-and-deploy` si occupa inizialmente di calcolare la nuova versione da associare alla libreria per poi inizializzare un repository di stage su Maven Central. Successivamente viene lanciato il *task* `gradle build` utilizzando una matrice con i sistemi operativi `windows-2022`, `macos-12`, `ubuntu-22.04`. Ciò permette di compilare il codice sorgente, eseguire i test, effettuare i controlli di code linting e bug detection e generare gli artefatti in ciascun ambiente. Questi sono generati sulla base dei *target* definiti nella lista al paragrafo 4.3; ciascun ambiente di compilazione, a seconda del sistema operativo in esecuzione, genera un *subset* di artefatti e li carica sul repository di stage. Infine il repository di stage viene chiuso e finalizzato. La finalizzazione del repository porta all'effettiva pubblicazione su Maven Central e si prosegue a pubblicare gli artefatti anche su NPM e GH Packages. Completato questo flusso si prosegue ad invocare il *workflow* `publish-docs`.

### 4.3.2 Generazione e pubblicazione della documentazione

Il *workflow* `publish-docs` provvede alla generazione della documentazione, sulla base del codice presente nel branch `master`.
Al termine del *workflow* `build-and-deploy` viene generato un *tag* associato al *merge commit* effettuato sul branch `master`. Questo *tag* rappresenta il nuovo valore della versione appena rilasciata, il quale viene riportato nelle pagine di documentazione generate attraverso il *task* `gradle dokkaHtml`. Queste pagine compongono un sito web statico e vengono copiate nel branch `gh-pages` grazie ad un'apposita *action*[3]. La presenza di una modifica in questo branch scatena l'esecuzione di un *workflow* di default, che mette a disposizione il contenuto tramite un webserver, accessibile all'indirizzo https://corinz97.github.io/EleKtion/ . In figura 4.3 viene mostrato come appare il sito web statico al termine delle operazioni.

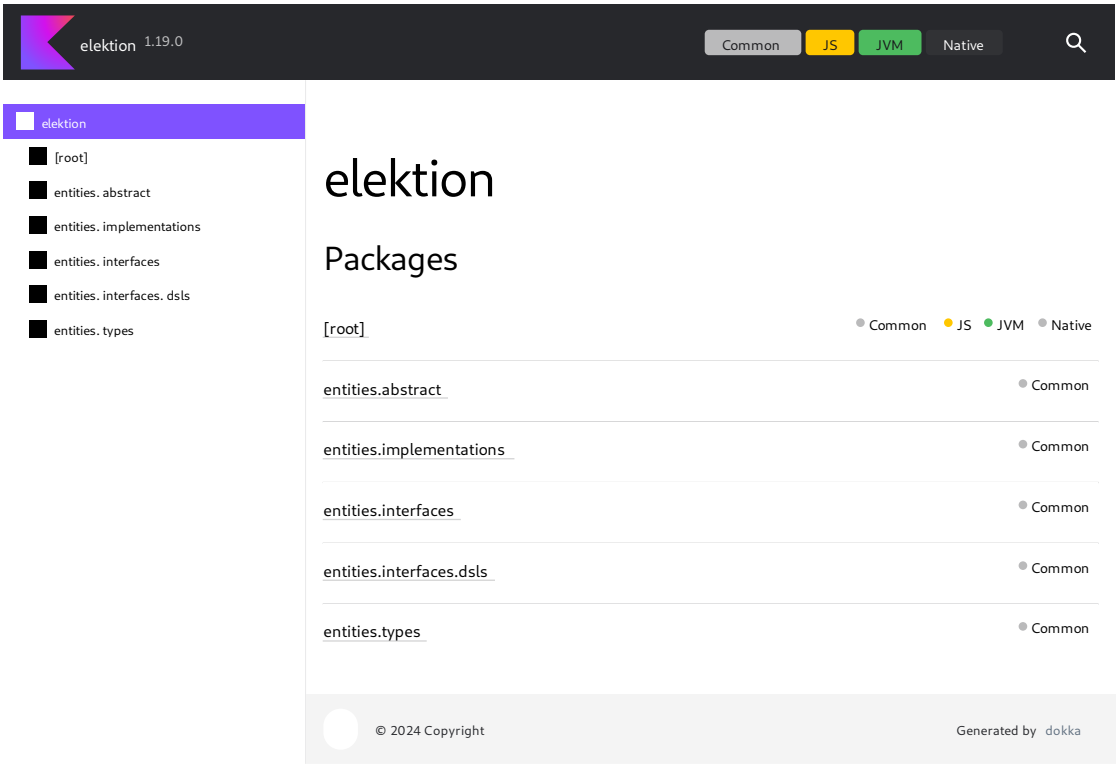


Figura 4.3: Sito web della documentazione, reso disponibile tramite GH Pages

[3] `https://archive.is/NKNH0`

# Capitolo 5

# Sperimentazione con i dati della Formula 1

La sperimentazione della libreria EleKtion è stata estesa ad un caso di studio reale, applicando gli algoritmi di votazione ai risultati delle gare di F1. I dati sono accessibili attraverso un API REST pubblica chiamata Ergast API[1], creata per scopi non-commerciali, che mette a disposizione dati inerenti ai mondiali di F1 a partire dall'anno 1950. Per ognuno di questi sono disponibili numerose informazioni come l'elenco delle gare ufficiali (Gran Premi), le gare di qualifica, le classifiche dei piloti, lo storico dei piloti, le classifiche dei costruttori e molto altro. Allo scopo di questa tesi, sono utili le informazioni riguardo ai risultati delle gare ufficiali e le classifiche provvisorie dei piloti al termine di ogni gara, entrambe legate ad uno specifico anno. Attraverso appositi endpoint, è possibile ottenere i dati per predisporre le informazioni necessarie ad EleKtion, utilizzando gli esiti delle varie gare come voti e applicando di conseguenza gli algoritmi a lista di preferenze. L'elemento più favorito nel voto è rappresentato dal pilota che si è classificato primo in gara, mentre l'elemento più sfavorito è il pilota classificatosi ultimo in gara. Nei seguenti paragrafi vengono mostrati i passaggi effettuati per elaborare i dati di un mondiale, in particolare vengono generate, come output delle votazioni, le classifiche provvisorie al termine di ogni gara in modo da avere uno storico, che verrà

[1] `https://archive.is/Iujgc`

confrontato con lo storico delle classifiche reali. Verrà infine svolto un calcolo utile ad indicare la differenza, in termini qualitativi, confrontando i risultati ottenuti.

## 5.1 Elaborazione dati di un mondiale F1

Si è provveduto ad ottenere l'elenco delle gare svoltesi nel corso del mondiale AAAA (anno del mondiale), contattando l'endpoint ergast.com/api/f1/AAAA.json. In questo mondiale sono state effettuate in totale $k$ gare, per ognuna di esse sono stati scaricati i relativi dati, filtrandoli, tenendo in considerazione solamente il nominativo del pilota e la posizione di arrivo. Eventuali ritiri a causa di guasti o altri motivi non sono gestiti, ciascun pilota in ogni caso ha attribuita una posizione di arrivo. I dati sono associati ad una mappa `Map<String, List<Pair<String, Int>>>`, nella quale le chiavi rappresentano il nome della gara (es. Australian-Grand-Prix-1-AAAA, Malaysian-Grand-Prix-2-AAAA, ecc...) mentre i valori rappresentano liste di tuple contenenti il nominativo del pilota e la posizione di arrivo. Come mostrato nell'estratto di codice 5.1, si organizzano tante votazioni indipendenti quante sono le gare presenti nel mondiale: viene definito l'insieme dei nominativi presenti nel mondiale, in modo da avere il censimento di tutti coloro che hanno partecipato almeno in una gara. Si definisce l'algoritmo da utilizzare, e in questo caso di studio sono ammissibili l'algoritmo di Condorcet o l'algoritmo di Schultze. Successivamente si determina l'elenco dei voti, in cui i votati sono costituiti dall'elenco dei nominativi dei piloti secondo l'ordine di arrivo reale (dal primo all'ultimo), e come votante viene utilizzato il nome del circuito della gara. Se uno o più piloti sono assenti in quella gara ma presenti in altre, tramite un confronto con l'insieme dei candidati vengono identificati e posti "virtualmente" ad ultimi in modo da avere la lunghezza della lista di preferenze uniforme al numero dei candidati. Infine, questa votazione viene inserita nell'elenco delle votazioni da elaborare. Per la seconda votazione viene considerata anche la seconda gara, e mantenendo stabile l'insieme dei candidati e l'algoritmo selezionato, vengono considerati come voti l'elenco dei nominativi (secondo l'ordine di arrivo reale) della prima gara e l'elenco dei nominativi (secondo l'ordine di arrivo reale) della seconda

gara. Proseguendo nelle iterazioni, si arriva ad avere $k$ votazioni indipendenti, in cui l'ultima conterrà come voti gli esiti dalla prima alla $k$-esima gara. La generazione delle classifiche provvisorie fornisce una serie di risultati in cui il primo consiste nella classifica provvisoria al termine della prima gara, il secondo nella classifica provvisoria al termine della seconda gara, fino ad arrivare alla classifica finale. I dati delle classifiche provvisorie reali sono ottenuti contattando l'endpoint ergast.com/api/f1/AAAA/GP/driverStandings.json, dove AAAA è l'anno del mondiale e GP è il numero della gara

Estratto di codice 5.1: Generazione delle classifiche provvisorie utilizzando EleKtion

```
val raceResults : Map<String, List<Pair<String, Int>>>() = ...
val validCompetitors = raceResults.flatMap { it.value }.groupBy({ it.first }, { it.second })
val allCompetitorNames = validCompetitors.keys.fold(setOf<String>()) { s, element -> s + element }
var election =
    PollManagerInstance<Nothing, ListOfPreferencesVote<Nothing>>() initializedAs {
        var currentGPs: Map<String, List<Pair<String, Int>>> = mapOf()
        var index = 1
        for (raceResult in raceResults) {
            currentGPs = currentGPs + raceResult.toPair()
            +poll {
                -competition("F1 Pilots - Temporary Ranking - GPs #1 to #${index++}") {
                    allCompetitorNames.forEach {
                        +competitor(it) {}
                    }
                }
                -condorcetAlgorithm{} // or -schultzeAlgorithm
                currentGPs.entries.forEach { (runningField, competitors) ->
                    +(competitors.fold(listOf<String>()) { l, element -> l then element.first }
                        votedBy runningField
                    )
                }
            }
        }
    }
var rankings = election.computeAllPolls()
```

Per confrontare gli esiti ottenuti impiegando i differenti algoritmi e quelli reali

estratti dall'API, è necessario organizzarli secondo matrici $r$ $x$ $k$, dove ogni riga $r$ identifica un pilota, e ogni colonna $k$ identifica una gara. Nell'intersezione è presente la posizione nella classifica provvisoria (reale o generata) ottenuta dal $r$-esimo pilota al termine della $k$-esima gara. Per effettuare un confronto qualitativo tra matrici è fondamentale organizzarle in modo tale che le righe e le colonne siano ordinate allo stesso modo. Nel caso delle righe è possibile adottare un ordinamento lessicografico o l'ordinamento presente in una delle matrici, che viene preso come riferimento per le altre. Nella sperimentazione effettuata, sono state calcolate per prime le classifiche che impiegano l'algoritmo di Condorcet, poi la sequenza dei piloti nella classifica finale è stato preso come riferimento di ordinamento. Le classifiche ottenute con l'algoritmo di Schultze sono state organizzate secondo lo stesso riferimento e così anche le classifiche reali. Si rimanda agli estratti di codice 5.2, 5.3, 5.4 per approfondire l'implementazione di questo ordinamento. Esistono numerose metriche di confronto tra matrici [GL96], in questo caso si è optato per la *norma di Frobenius*, rappresentata dalla formula $\sqrt{\sum_{i,j} abs(a_{i,j})^2}$, dove $a_{i,j}$ è l'elemento di una matrice $A$ ricavata dalla differenza tra le matrici trattate.

Estratto di codice 5.2: Ordinamento propedeutico al confronto (Condorcet)

```
val flattenedOrdersInRankings = rankings.map { it.ranking.flatMap { it.key.map {
    it.name } } }
val condorcetFlattenedOrderInFinalRanking = flattenedOrdersInRankings.last()
var m: Map<String, List<Int>> = mapOf()
for (competitor in condorcetFlattenedOrderInFinalRanking) {
    var listOfPositionsPerCompetitor: List<Int> = listOf()
    for (ranking in rankings.map { it.ranking.keys.toList() }) {
        var index = ranking.indexOfFirst { it.map { it.name }.contains(competitor)
             }
        ++index
        listOfPositionsPerCompetitor = listOfPositionsPerCompetitor + index
    }
    m = m + (competitor to listOfPositionsPerCompetitor)
}
```

Estratto di codice 5.3: Ordinamento propedeutico al confronto (Schultze)

```
val schultzeFlattenedOrderInFinalRanking = condorcetFlattenedOrderInFinalRanking
var m: Map<String, List<Int>> = mapOf()
for (competitor in schultzeFlattenedOrderInFinalRanking) {
    var listOfPositionsPerCompetitor: List<Int> = listOf()
```

```
    for (ranking in rankings.map { it.ranking.keys.toList() }) {
        var index = ranking.indexOfFirst { it.map { it.name }.contains(competitor)
            }
        ++index
        listOfPositionsPerCompetitor = listOfPositionsPerCompetitor + index
    }
    m = m + (competitor to listOfPositionsPerCompetitor)
}
```

Estratto di codice 5.4: Ordinamento propedeutico al confronto (Dati reali)

```
val realWorldFlattenedOrderInFinalRanking = condorcetFlattenedOrderInFinalRanking
var m: Map<String, List<Int>> = mapOf()
for (competitor in realWorldFlattenedOrderInFinalRanking) {
    var listOfPositionsPerCompetitor: List<Int> = listOf()

    for (ranking in raceResults.map { it.value }) {
        val index = ranking.firstOrNull { it.first == competitor }?.second
            ?: ranking.maxOf { it.second }
        listOfPositionsPerCompetitor = listOfPositionsPerCompetitor + index
    }
    m = m + (competitor to listOfPositionsPerCompetitor)
}
```

## 5.2 Elaborazione del mondiale F1 2023

Si è provveduto alla generazione delle classifiche provvisorie utilizzando i dati del mondiale F1 2023. Si riporta in tabella 5.1 la classifica finale reale e quelle finali basate sull'algoritmo di Condorcet e l'algoritmo di Schultze.

| Posizione | Reale | Condorcet | Schultze |
|---|---|---|---|
| 1. | Max-Verstappen | Max-Verstappen (C1.) | Max-Verstappen |
| 2. | Sergio-Pérez | Sergio-Pérez (C2.) | Sergio-Pérez |
| 3. | Lewis-Hamilton | Lewis-Hamilton (C3.) | Lewis-Hamilton |
| 4. | Fernando-Alonso | Fernando-Alonso (C4.)<br>Carlos-Sainz (C5.)<br>Charles-Leclerc (C6.) | Fernando-Alonso |
| 5. | Charles-Leclerc | Lando-Norris (C7.) | Carlos-Sainz |
| 6. | Lando-Norris | George-Russell (C8.) | Lando-Norris |
| 7. | Carlos-Sainz | Lance-Stroll (C9.)<br>Oscar-Piastri (C10.) | George-Russell |
| 8. | George-Russell | Pierre-Gasly (C11.) | Charles-Leclerc |
| 9. | Oscar-Piastri | Alexander-Albon (C12.)<br>Esteban-Ocon (C13.) | Pierre-Gasly |
| 10. | Lance-Stroll | Yuki-Tsunoda (C14.) | Oscar-Piastri |
| 11. | Pierre-Gasly | Valtteri-Bottas (C15.)<br>Nico-Hülkenberg (C16.) | Lance-Stroll |
| 12. | Esteban-Ocon | Guanyu-Zhou (C17.) | Esteban-Ocon |
| 13. | Alexander-Albon | Kevin-Magnussen (C18.) | Alexander-Albon |
| 14. | Yuki-Tsunoda | Logan-Sargeant (C19.) | Yuki-Tsunoda |
| 15. | Valtteri-Bottas | Nyck-de Vries (C20.) | Valtteri-Bottas |
| 16. | Nico-Hülkenberg | Daniel-Ricciardo (C21.) | Guanyu-Zhou |
| 17. | Daniel-Ricciardo | Liam-Lawson (C22.) | Nico-Hülkenberg |
| 18. | Guanyu-Zhou | Assente | Kevin-Magnussen |
| 19. | Kevin-Magnussen | Assente | Logan-Sargeant |
| 20. | Liam-Lawson | Assente | Daniel-Ricciardo |
| 21. | Logan-Sargeant | Assente | Nyck-de Vries |
| 22. | Nyck-de Vries | Assente | Liam-Lawson |

Tabella 5.1: Classifica finale reale e classifiche finali ottenute con l'algoritmo di Condorcet e l'algoritmo di Schultze, relativamente al mondiale F1 2023. Nella colonna relativa all'algoritmo di Condorcet viene fornito un acronimo ad ogni concorrente.

Nelle tabelle 5.2, 5.3 e 5.4 vengono mostrate le matrici complete con il posizionamento dei piloti in ogni classifica provvisoria. Per motivi prettamente grafici, a

ciascun concorrente è stato assegnato un acronimo, consultabile nella terza colonna della tabella 5.1 . Infine, per ciascuna tabella è stato generato il relativo grafico, nel quale le linee rappresentano i concorrenti, l'asse delle ascisse rappresenta il numero sequenziale della gara e l'asse delle ordinate è associato al posizionamento in classifica al termine della relativa gara. Per la tabella 5.2 il grafico di riferimento è in figura 5.1, mentre per le tabelle 5.3 e 5.4 i grafici sono rispettivamente in figura 5.2 ed in figura 5.3.

| | G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 | G10 | G11 | G12 | G13 | G14 | G15 | G16 | G17 | G18 | G19 | G20 | G21 | G22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1. | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| C2. | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| C3. | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| C4. | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 5 | 4 | 5 | 4 |
| C5. | 4 | 4 | 5 | 5 | 5 | 6 | 6 | 5 | 5 | 5 | 6 | 7 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 6 | 4 | 7 |
| C6. | 19 | 8 | 10 | 6 | 7 | 7 | 7 | 7 | 6 | 7 | 7 | 5 | 6 | 6 | 6 | 6 | 6 | 7 | 7 | 7 | 7 | 5 |
| C7. | 17 | 20 | 8 | 9 | 9 | 11 | 11 | 11 | 10 | 9 | 8 | 8 | 8 | 8 | 8 | 7 | 7 | 6 | 6 | 5 | 6 | 6 |
| C8. | 7 | 6 | 7 | 7 | 6 | 5 | 5 | 6 | 7 | 6 | 5 | 6 | 7 | 7 | 7 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| C9. | 6 | 7 | 6 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 9 | 9 | 9 | 9 | 9 | 10 | 10 | 11 | 11 | 10 | 10 | 10 |
| C10. | 20 | 19 | 13 | 11 | 14 | 13 | 13 | 14 | 14 | 11 | 11 | 11 | 12 | 12 | 11 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| C11. | 9 | 11 | 14 | 14 | 10 | 10 | 10 | 10 | 11 | 12 | 12 | 12 | 10 | 10 | 10 | 11 | 11 | 10 | 10 | 11 | 11 | 11 |
| C12. | 10 | 13 | 18 | 17 | 18 | 18 | 18 | 12 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| C13. | 18 | 10 | 12 | 13 | 12 | 9 | 9 | 9 | 9 | 10 | 10 | 10 | 11 | 11 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| C14. | 11 | 14 | 16 | 16 | 16 | 16 | 16 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 16 | 16 | 14 | 14 | 14 |
| C15. | 8 | 9 | 11 | 12 | 13 | 14 | 14 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 14 | 14 | 14 | 15 | 15 | 15 |
| C16. | 15 | 15 | 9 | 10 | 11 | 12 | 12 | 13 | 12 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 15 | 15 | 15 | 16 | 16 | 16 |
| C17. | 16 | 17 | 15 | 15 | 15 | 15 | 15 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 17 | 18 | 18 | 18 | 18 |
| C18. | 13 | 12 | 17 | 18 | 17 | 17 | 17 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 19 | 19 | 19 | 19 |
| C19. | 12 | 16 | 19 | 19 | 19 | 20 | 20 | 20 | 19 | 19 | 19 | 19 | 19 | 19 | 20 | 20 | 20 | 20 | 21 | 21 | 21 | 21 |
| C20. | 14 | 18 | 20 | 20 | 20 | 19 | 19 | 19 | 20 | 20 | 20 | 20 | 20 | 21 | 21 | 21 | 21 | 21 | 22 | 22 | 22 | 22 |
| C21. | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 21 | 21 | 21 | 22 | 22 | 22 | 22 | 22 | 17 | 17 | 17 | 17 |
| C22. | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 21 | 21 | 22 | 20 | 19 | 19 | 19 | 19 | 20 | 20 | 20 | 20 |

Tabella 5.2: Tabella relativa ai valori reali del Mondiale F1 2023, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica generale, al termine della gara stessa. I valori sono ricavati da Ergast API .

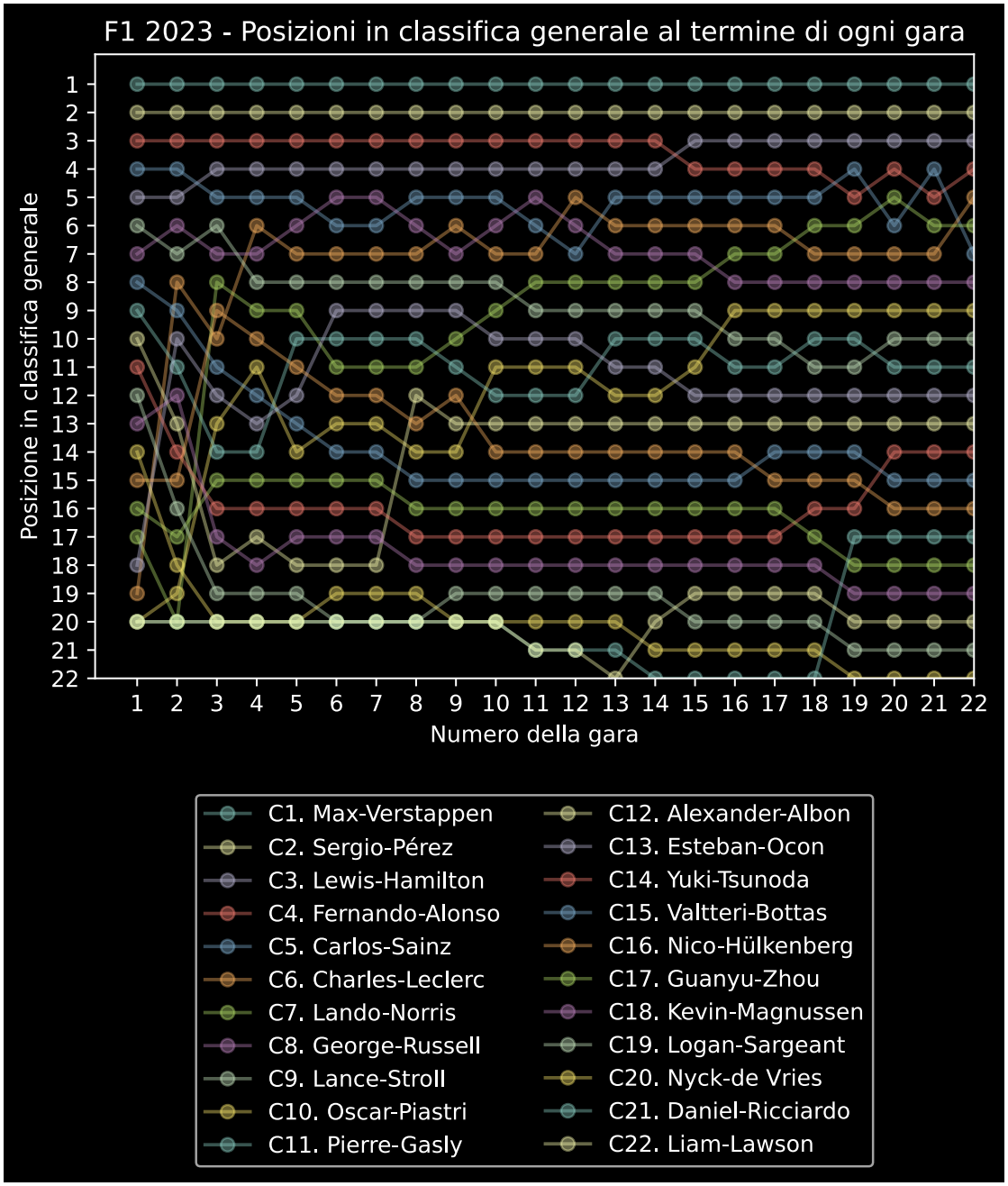


Figura 5.1: Grafico a linee in cui sono utilizzati i valori presenti nella tabella 5.2

| | G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 | G10 | G11 | G12 | G13 | G14 | G15 | G16 | G17 | G18 | G19 | G20 | G21 | G22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1. | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| C2. | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| C3. | 5 | 3 | 4 | 3 | 5 | 4 | 4 | 3 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 3 | 4 | 3 |
| C4. | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 |
| C5. | 4 | 3 | 5 | 3 | 4 | 5 | 6 | 4 | 5 | 5 | 7 | 7 | 7 | 6 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 4 |
| C6. | 19 | 10 | 14 | 11 | 6 | 7 | 7 | 6 | 7 | 5 | 6 | 6 | 6 | 6 | 7 | 5 | 5 | 7 | 5 | 6 | 5 | 4 |
| C7. | 17 | 11 | 13 | 7 | 11 | 10 | 11 | 10 | 11 | 8 | 10 | 9 | 8 | 7 | 8 | 6 | 6 | 6 | 6 | 5 | 6 | 5 |
| C8. | 7 | 4 | 7 | 4 | 6 | 6 | 5 | 5 | 6 | 4 | 5 | 5 | 5 | 5 | 6 | 5 | 7 | 6 | 7 | 5 | 7 | 6 |
| C9. | 6 | 11 | 6 | 4 | 6 | 9 | 7 | 7 | 8 | 6 | 8 | 8 | 8 | 7 | 9 | 9 | 9 | 10 | 9 | 9 | 8 | 7 |
| C10. | 20 | 11 | 14 | 8 | 15 | 12 | 12 | 10 | 13 | 10 | 11 | 12 | 9 | 10 | 11 | 8 | 8 | 9 | 8 | 8 | 9 | 7 |
| C11. | 9 | 5 | 8 | 6 | 7 | 8 | 8 | 8 | 9 | 7 | 9 | 10 | 8 | 8 | 10 | 7 | 8 | 8 | 8 | 7 | 8 | 8 |
| C12. | 10 | 11 | 14 | 11 | 13 | 13 | 16 | 12 | 14 | 11 | 11 | 13 | 9 | 9 | 11 | 8 | 9 | 10 | 9 | 9 | 10 | 9 |
| C13. | 18 | 10 | 11 | 9 | 9 | 9 | 9 | 8 | 10 | 9 | 11 | 11 | 9 | 9 | 11 | 9 | 9 | 11 | 9 | 10 | 11 | 9 |
| C14. | 11 | 6 | 8 | 5 | 8 | 9 | 10 | 9 | 12 | 12 | 13 | 13 | 11 | 12 | 13 | 11 | 11 | 13 | 11 | 11 | 13 | 10 |
| C15. | 8 | 10 | 11 | 9 | 12 | 11 | 13 | 11 | 13 | 11 | 12 | 13 | 10 | 11 | 12 | 10 | 10 | 12 | 10 | 12 | 12 | 11 |
| C16. | 15 | 8 | 9 | 7 | 10 | 11 | 11 | 12 | 15 | 13 | 14 | 14 | 12 | 13 | 14 | 12 | 12 | 14 | 12 | 12 | 14 | 11 |
| C17. | 16 | 9 | 10 | 10 | 13 | 12 | 14 | 13 | 15 | 13 | 15 | 14 | 13 | 14 | 15 | 13 | 13 | 15 | 13 | 13 | 15 | 12 |
| C18. | 13 | 7 | 12 | 9 | 9 | 12 | 13 | 13 | 15 | 15 | 16 | 15 | 14 | 15 | 16 | 14 | 14 | 16 | 14 | 14 | 16 | 13 |
| C19. | 12 | 10 | 13 | 11 | 16 | 14 | 17 | 15 | 17 | 15 | 18 | 16 | 15 | 16 | 17 | 15 | 15 | 17 | 15 | 15 | 17 | 14 |
| C20. | 14 | 9 | 12 | 11 | 14 | 13 | 15 | 14 | 16 | 14 | 17 | 16 | 16 | 17 | 18 | 16 | 16 | 18 | 16 | 16 | 18 | 15 |
| C21. | 21 | 12 | 15 | 12 | 17 | 15 | 18 | 16 | 18 | 16 | 19 | 17 | 17 | 18 | 19 | 17 | 17 | 19 | 17 | 17 | 19 | 16 |
| C22. | 22 | 13 | 16 | 13 | 18 | 16 | 19 | 17 | 19 | 17 | 20 | 18 | 18 | 19 | 20 | 18 | 18 | 20 | 18 | 18 | 20 | 17 |

Tabella 5.3: Tabella relativa al Mondiale F1 2023, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica, al termine della gara stessa. I valori sono ricavati dall'applicazione dell'algoritmo di Condorcet al termine di ogni gara, considerando anche le precedenti. Ogni gara rappresenta un voto, composto dagli ordini di arrivo.

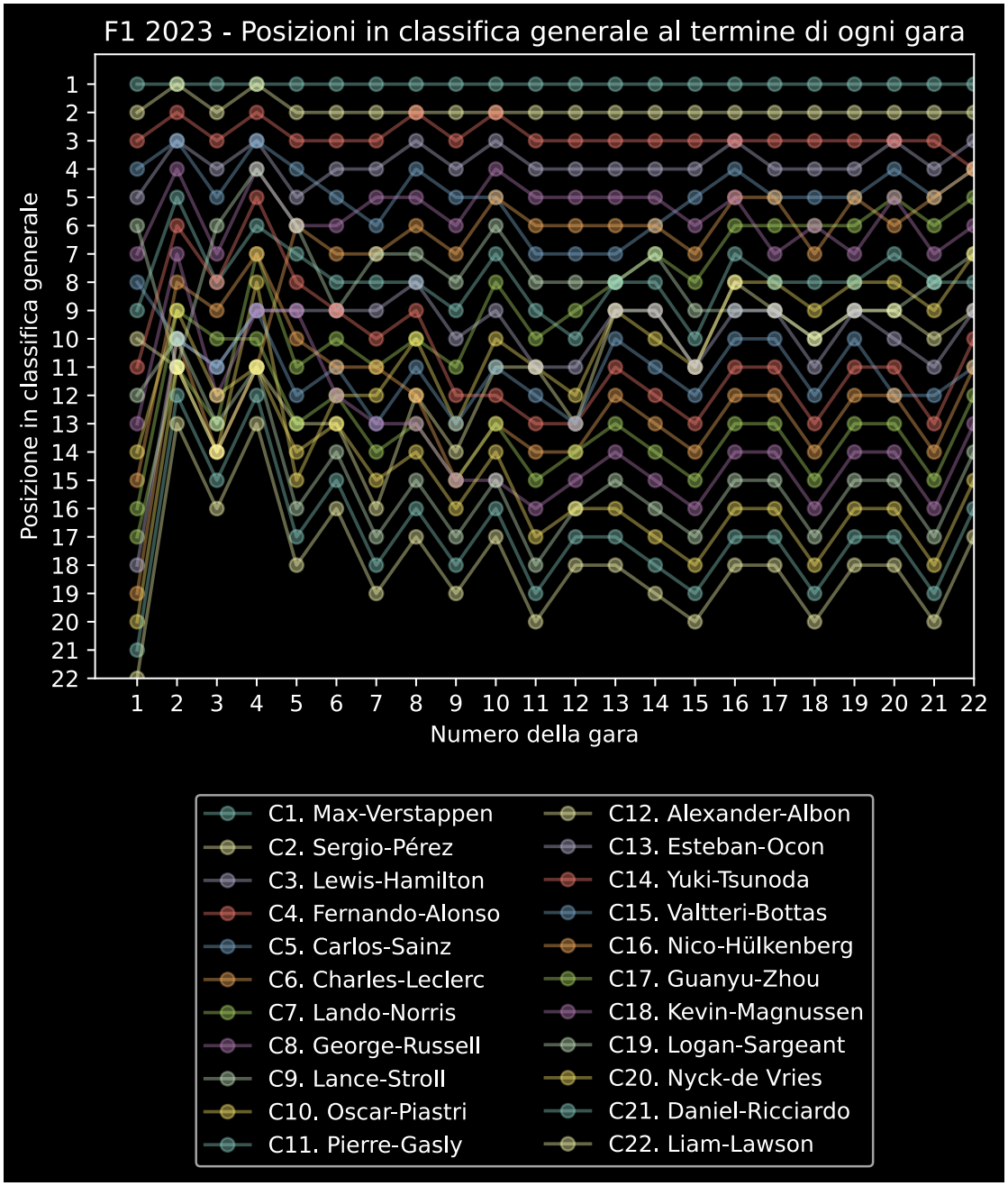


Figura 5.2: Grafico a linee in cui sono utilizzati i valori presenti nella tabella 5.3

| | G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 | G10 | G11 | G12 | G13 | G14 | G15 | G16 | G17 | G18 | G19 | G20 | G21 | G22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1. | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| C2. | 2 | 1 | 2 | 2 | 2 | 3 | 4 | 4 | 3 | 4 | 3 | 2 | 2 | 2 | 2 | 3 | 4 | 2 | 3 | 3 | 2 | 2 |
| C3. | 5 | 3 | 4 | 4 | 4 | 4 | 3 | 3 | 4 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 2 | 4 | 2 | 2 | 3 | 3 |
| C4. | 3 | 2 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 2 | 3 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 4 |
| C5. | 4 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 6 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| C6. | 19 | 8 | 16 | 9 | 9 | 8 | 8 | 7 | 7 | 7 | 6 | 7 | 6 | 6 | 6 | 6 | 6 | 8 | 8 | 8 | 8 | 8 |
| C7. | 17 | 12 | 13 | 9 | 12 | 12 | 11 | 12 | 11 | 10 | 7 | 8 | 7 | 7 | 8 | 8 | 8 | 6 | 6 | 6 | 6 | 6 |
| C8. | 7 | 4 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| C9. | 6 | 8 | 7 | 6 | 7 | 11 | 9 | 10 | 9 | 8 | 8 | 9 | 8 | 8 | 10 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| C10. | 20 | 13 | 14 | 12 | 14 | 15 | 13 | 13 | 14 | 12 | 10 | 12 | 11 | 11 | 11 | 10 | 9 | 10 | 10 | 10 | 10 | 10 |
| C11. | 9 | 5 | 8 | 8 | 8 | 7 | 7 | 8 | 8 | 9 | 9 | 10 | 9 | 9 | 9 | 9 | 10 | 9 | 9 | 9 | 9 | 9 |
| C12. | 10 | 11 | 17 | 15 | 15 | 17 | 16 | 15 | 13 | 14 | 12 | 14 | 12 | 10 | 12 | 12 | 13 | 12 | 12 | 13 | 13 | 13 |
| C13. | 18 | 8 | 13 | 13 | 11 | 9 | 9 | 9 | 10 | 11 | 11 | 11 | 10 | 11 | 13 | 13 | 12 | 13 | 13 | 12 | 12 | 12 |
| C14. | 11 | 6 | 9 | 7 | 8 | 10 | 10 | 11 | 12 | 13 | 13 | 13 | 13 | 13 | 15 | 14 | 16 | 15 | 14 | 14 | 14 | 14 |
| C15. | 8 | 8 | 11 | 13 | 13 | 13 | 14 | 14 | 14 | 15 | 14 | 15 | 14 | 12 | 14 | 14 | 14 | 14 | 14 | 15 | 15 | 15 |
| C16. | 15 | 9 | 10 | 10 | 12 | 15 | 14 | 16 | 15 | 17 | 16 | 17 | 16 | 15 | 17 | 16 | 17 | 17 | 16 | 17 | 17 | 17 |
| C17. | 16 | 11 | 12 | 14 | 14 | 16 | 12 | 15 | 14 | 16 | 15 | 16 | 15 | 14 | 16 | 15 | 15 | 16 | 15 | 16 | 16 | 16 |
| C18. | 13 | 7 | 13 | 11 | 10 | 14 | 15 | 17 | 16 | 18 | 17 | 18 | 17 | 16 | 18 | 17 | 18 | 18 | 17 | 18 | 18 | 18 |
| C19. | 12 | 10 | 15 | 15 | 16 | 19 | 18 | 19 | 18 | 20 | 18 | 19 | 18 | 17 | 19 | 18 | 19 | 19 | 18 | 19 | 19 | 19 |
| C20. | 14 | 10 | 14 | 16 | 17 | 18 | 17 | 18 | 17 | 19 | 18 | 20 | 19 | 18 | 20 | 19 | 20 | 20 | 19 | 20 | 20 | 21 |
| C21. | 21 | 14 | 18 | 17 | 18 | 20 | 19 | 20 | 19 | 21 | 19 | 21 | 20 | 19 | 22 | 21 | 22 | 22 | 21 | 21 | 21 | 20 |
| C22. | 22 | 15 | 19 | 18 | 19 | 21 | 20 | 21 | 20 | 22 | 20 | 22 | 21 | 20 | 21 | 20 | 21 | 21 | 20 | 22 | 22 | 22 |

Tabella 5.4: Tabella relativa al Mondiale F1 2023, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica, al termine della gara stessa. I valori sono ricavati dall'applicazione dell'algoritmo di Schultze al termine di ogni gara, considerando anche le precedenti. Ogni gara rappresenta un voto, composto dagli ordini di arrivo.

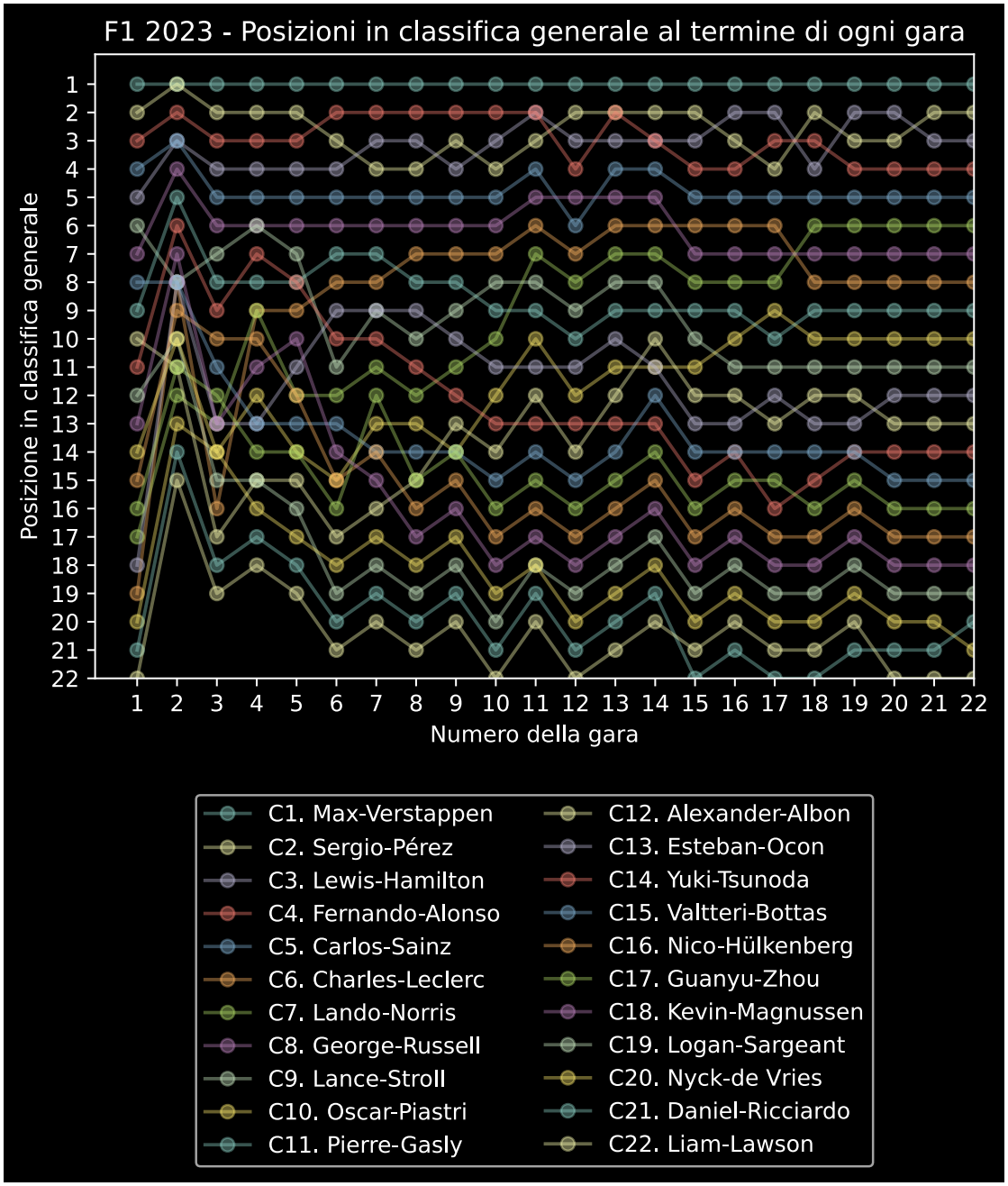


Figura 5.3: Grafico a linee in cui sono utilizzati i valori presenti nella tabella 5.4

Si può notare che entrambi gli algoritmi impiegati tendono ad una soluzione che porta i concorrenti ad essere molto simili come posizionamento nelle prime

fasi del campionato, causando numerosi pareggi ma proseguendo con le gare questi tendono a ridursi per via del numero incrementale di voti. Tuttavia, i risultati evidenziati da Condorcet nell'ultima classifica manifestano numerosi pareggi, mentre Schultze riesce a discriminare in maniera più accurata gli esiti. In tabella 5.5 vengono mostrati i valori di distanza ottenuti applicando la *norma di Frobenius* alla differenza delle matrici rappresentate nelle prime due colonne. Si può notare che l'insieme delle classifiche generate sono significativamente diverse dall'insieme delle classifiche reali, mentre se ci si focalizza sulle classifiche finali la distanza è minore ma i valori generati da Condorcet continuano a distinguersi maggiormente, soprattutto a causa dei pareggi, che nella classifica finale reale non sono presenti.

| Elemento 1 | Elemento 2 | Valore distanza euclidea |
|---|---|---|
| Matrice "reale" | Matrice Condorcet | 68.23 |
| Matrice "reale" | Matrice Schultze | 44.72 |
| Classifica finale "reale" | Classifica finale Condorcet | 14.35 |
| Classifica finale "reale" | Classifica finale Schultze | 8.31 |

Tabella 5.5: Campionato F1 2023 - Valori di distanza ottenuti applicando la *norma di Frobenius* alla differenza delle matrici rappresentate nelle prime due colonne.

## 5.3 Elaborazione del mondiale F1 2008

È stata effettuata l'elaborazione del mondiale F1 2008, che rispetto al mondiale 2023 ha visto una maggior competizione tra i concorrenti per le prime posizioni. Si è provveduto a generare le classifiche provvisorie replicando il flusso rappresentato nel paragrafo 5.1. Si riporta in tabella 5.6 la classifica finale reale ed i risultati ottenuti nelle classifiche finali, usando l'algoritmo di Condorcet e l'algoritmo di Schultze.

| Posizione | Reale | Condorcet | Schultze |
|---|---|---|---|
| 1. | Lewis-Hamilton | Felipe-Massa (C1.) | Lewis-Hamilton |
| 2. | Felipe-Massa | Lewis-Hamilton (C2.) | Robert-Kubica |
| 3. | Kimi-Räikkönen | Kimi-Räikkönen (C3.)<br>Robert-Kubica (C4.) | Felipe-Massa |
| 4. | Robert-Kubica | Nick-Heidfeld (C5.) | Nick-Heidfeld |
| 5. | Fernando-Alonso | Heikki-Kovalainen (C6.) | Kimi-Räikkönen |
| 6. | Nick-Heidfeld | Fernando-Alonso (C7.) | Fernando-Alonso |
| 7. | Heikki-Kovalainen | Jarno-Trulli (C8.) | Heikki-Kovalainen |
| 8. | Sebastian-Vettel | Timo-Glock (C9.)<br>Mark-Webber (C10.) | Jarno-Trulli |
| 9. | Jarno-Trulli | Nico-Rosberg (C11.)<br>Sebastian-Vettel (C12.) | Mark-Webber |
| 10. | Timo-Glock | David-Coulthard (C13.) | Timo-Glock |
| 11. | Mark-Webber | Kazuki-Nakajima (C14.)<br>Rubens-Barrichello (C15.) | Nico-Rosberg |
| 12. | Nelson-Piquet Jr. | Sébastien-Bourdais (C16.)<br>Nelson-Piquet Jr. (C17.)<br>Jenson-Button (C18.) | Sebastian-Vettel |
| 13. | Nico-Rosberg | Giancarlo-Fisichella (C19.) | Kazuki-Nakajima |
| 14. | Rubens-Barrichello | Adrian-Sutil (C20.) | Nelson-Piquet Jr. |
| 15. | Kazuki-Nakajima | Takuma-Sato (C21.) | David-Coulthard |
| 16. | David-Coulthard | Anthony-Davidson (C22.) | Jenson-Button |
| 17. | Sébastien-Bourdais | Assente | Rubens-Barrichello |
| 18. | Jenson-Button | Assente | Sébastien-Bourdais |
| 19. | Giancarlo-Fisichella | Assente | Giancarlo-Fisichella |
| 20. | Adrian-Sutil | Assente | Adrian-Sutil |
| 21. | Takuma-Sato | Assente | Takuma-Sato |
| 22. | Anthony-Davidson | Assente | Anthony-Davidson |

Tabella 5.6: Classifica finale reale e classifiche finali ottenute con l'algoritmo di Condorcet e l'algoritmo di Schultze, relativamente al mondiale F1 2008. Nella colonna relativa all'algoritmo di Condorcet viene fornito un acronimo ad ogni concorrente.

Nelle tabelle 5.7, 5.8 e 5.9 vengono mostrate le matrici complete con il posizio-

namento dei piloti in ogni classifica provvisoria. Per motivi prettamente grafici, a ciascun concorrente è stato assegnato un acronimo, consultabile nella terza colonna della tabella 5.6 Infine, per ciascuna tabella è stato generato il relativo grafico, nel quale le linee rappresentano i concorrenti, l'asse delle ascisse rappresenta il numero sequenziale della gara e l'asse delle ordinate è associato al posizionamento in classifica al termine della relativa gara. Per la tabella 5.7 il grafico di riferimento è in figura 5.4, mentre per le tabelle 5.8 e 5.9 i grafici sono rispettivamente in figura 5.5 ed in figura 5.6.

| | G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 | G10 | G11 | G12 | G13 | G14 | G15 | G16 | G17 | G18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1. | 8 | 18 | 6 | 4 | 2 | 3 | 3 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| C2. | 1 | 1 | 3 | 2 | 3 | 1 | 2 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| C3. | 8 | 2 | 1 | 1 | 1 | 2 | 4 | 3 | 3 | 3 | 2 | 3 | 4 | 4 | 4 | 4 | 4 | 3 |
| C4. | 8 | 5 | 4 | 3 | 4 | 4 | 1 | 2 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 4 |
| C5. | 2 | 3 | 2 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 6 | 5 | 5 | 5 | 5 | 5 | 6 |
| C6. | 5 | 4 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 5 | 6 | 6 | 6 | 6 | 7 | 7 |
| C7. | 4 | 7 | 9 | 10 | 8 | 8 | 9 | 9 | 9 | 9 | 8 | 8 | 8 | 7 | 7 | 7 | 6 | 5 |
| C8. | 8 | 8 | 7 | 7 | 9 | 9 | 8 | 7 | 7 | 7 | 7 | 7 | 7 | 8 | 9 | 9 | 9 | 9 |
| C9. | 8 | 18 | 14 | 14 | 15 | 17 | 13 | 13 | 14 | 16 | 10 | 10 | 10 | 11 | 10 | 11 | 10 | 10 |
| C10. | 8 | 11 | 10 | 8 | 7 | 7 | 7 | 8 | 8 | 8 | 9 | 9 | 9 | 10 | 11 | 10 | 11 | 11 |
| C11. | 3 | 6 | 8 | 9 | 10 | 10 | 10 | 10 | 11 | 12 | 13 | 13 | 14 | 14 | 12 | 13 | 13 | 13 |
| C12. | 8 | 18 | 21 | 21 | 22 | 12 | 14 | 14 | 15 | 15 | 16 | 14 | 12 | 9 | 8 | 8 | 8 | 8 |
| C13. | 8 | 12 | 13 | 15 | 14 | 16 | 12 | 12 | 13 | 14 | 15 | 16 | 16 | 16 | 16 | 16 | 16 | 16 |
| C14. | 6 | 9 | 11 | 11 | 11 | 11 | 11 | 11 | 12 | 13 | 14 | 15 | 15 | 15 | 15 | 15 | 15 | 15 |
| C15. | 8 | 16 | 16 | 17 | 17 | 14 | 15 | 15 | 10 | 10 | 12 | 12 | 13 | 13 | 14 | 14 | 14 | 14 |
| C16. | 7 | 10 | 12 | 13 | 13 | 15 | 17 | 18 | 17 | 18 | 18 | 18 | 17 | 17 | 17 | 17 | 17 | 17 |
| C17. | 8 | 14 | 17 | 18 | 18 | 19 | 19 | 17 | 18 | 11 | 11 | 11 | 11 | 12 | 13 | 12 | 12 | 12 |
| C18. | 8 | 13 | 15 | 12 | 12 | 13 | 16 | 16 | 16 | 17 | 17 | 17 | 18 | 18 | 18 | 18 | 18 | 18 |
| C19. | 8 | 15 | 18 | 16 | 16 | 18 | 18 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 | 19 |
| C20. | 8 | 18 | 21 | 21 | 21 | 22 | 22 | 22 | 22 | 21 | 21 | 21 | 20 | 20 | 20 | 20 | 20 | 20 |
| C21. | 8 | 18 | 20 | 19 | 19 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 21 | 21 | 21 | 21 | 21 | 21 |
| C22. | 8 | 17 | 19 | 20 | 20 | 21 | 21 | 21 | 21 | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 22 |

Tabella 5.7: Tabella relativa al Mondiale F1 2008, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica generale, al termine della gara stessa. I valori sono ricavati da Ergast API .

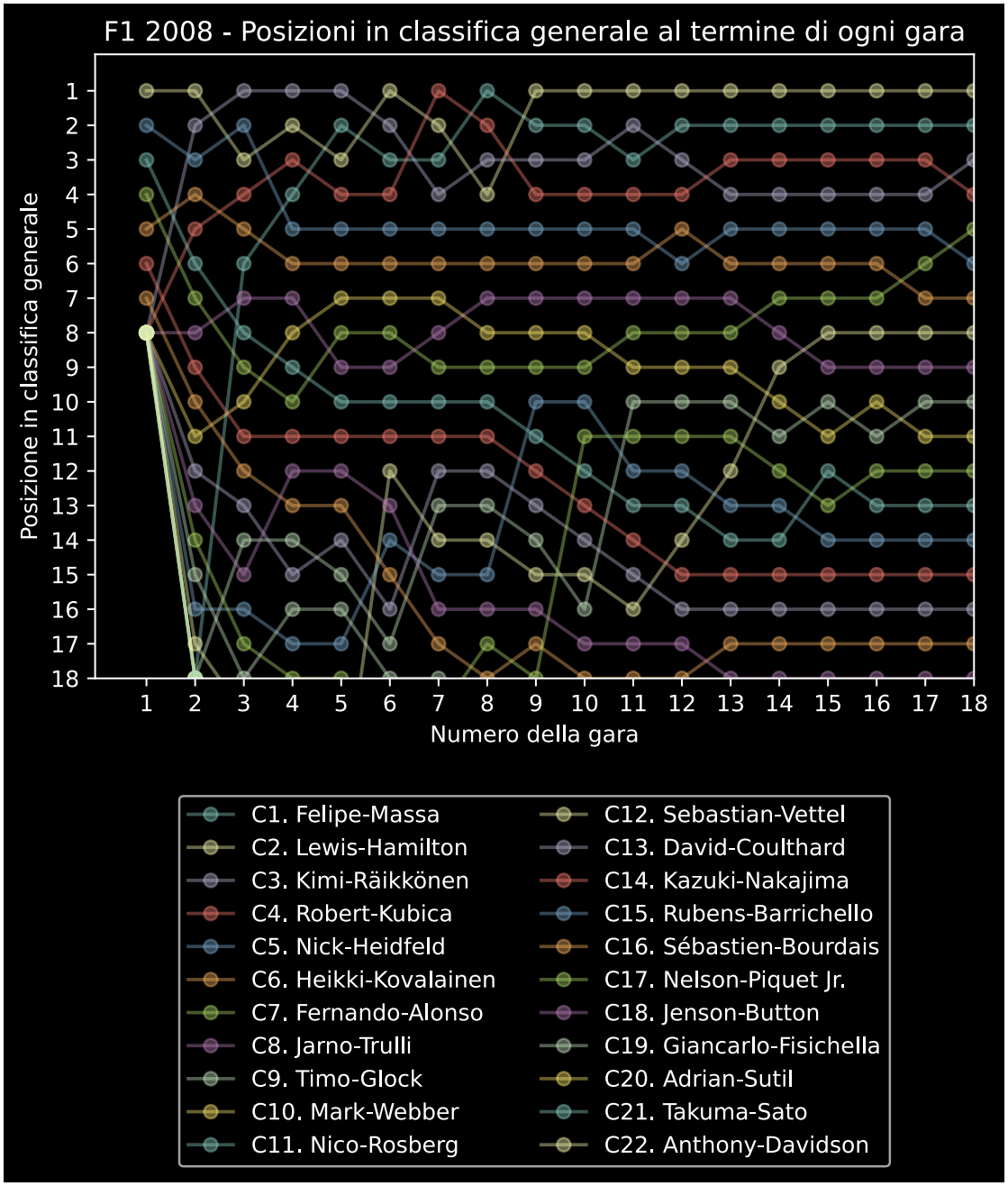


Figura 5.4: Grafico a linee in cui sono utilizzati i valori presenti nella tabella 5.7

| | G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 | G10 | G11 | G12 | G13 | G14 | G15 | G16 | G17 | G18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1. | 13 | 12 | 8 | 6 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| C2. | 1 | 1 | 4 | 3 | 3 | 1 | 2 | 3 | 3 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 2 |
| C3. | 8 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 3 | 3 | 5 | 5 | 4 | 3 | 3 |
| C4. | 9 | 5 | 2 | 2 | 4 | 2 | 3 | 2 | 4 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 3 | 3 |
| C5. | 2 | 2 | 3 | 3 | 5 | 3 | 4 | 3 | 5 | 4 | 3 | 4 | 4 | 4 | 3 | 3 | 4 | 4 |
| C6. | 5 | 3 | 3 | 4 | 9 | 5 | 5 | 4 | 6 | 5 | 4 | 4 | 5 | 4 | 4 | 3 | 5 | 5 |
| C7. | 4 | 5 | 6 | 5 | 8 | 6 | 9 | 6 | 9 | 8 | 7 | 7 | 8 | 7 | 6 | 5 | 6 | 6 |
| C8. | 15 | 7 | 4 | 4 | 6 | 6 | 6 | 4 | 7 | 6 | 5 | 5 | 6 | 6 | 7 | 6 | 7 | 7 |
| C9. | 10 | 13 | 7 | 7 | 11 | 8 | 11 | 8 | 11 | 10 | 9 | 9 | 9 | 8 | 9 | 8 | 9 | 8 |
| C10. | 17 | 9 | 5 | 4 | 7 | 4 | 7 | 5 | 8 | 7 | 6 | 6 | 7 | 7 | 8 | 7 | 8 | 8 |
| C11. | 3 | 6 | 5 | 6 | 7 | 7 | 8 | 7 | 10 | 9 | 8 | 8 | 10 | 9 | 10 | 9 | 10 | 9 |
| C12. | 20 | 13 | 16 | 13 | 19 | 14 | 19 | 14 | 16 | 13 | 13 | 13 | 13 | 11 | 12 | 10 | 10 | 9 |
| C13. | 14 | 8 | 9 | 8 | 12 | 9 | 12 | 7 | 12 | 10 | 10 | 10 | 11 | 10 | 11 | 11 | 11 | 10 |
| C14. | 6 | 9 | 7 | 7 | 11 | 8 | 11 | 10 | 11 | 10 | 11 | 12 | 12 | 11 | 12 | 12 | 12 | 11 |
| C15. | 22 | 13 | 12 | 9 | 13 | 10 | 13 | 8 | 13 | 10 | 11 | 11 | 12 | 13 | 14 | 14 | 12 | 11 |
| C16. | 7 | 13 | 9 | 10 | 14 | 13 | 16 | 12 | 15 | 13 | 13 | 13 | 13 | 13 | 14 | 14 | 13 | 12 |
| C17. | 12 | 8 | 10 | 11 | 16 | 11 | 14 | 11 | 15 | 12 | 12 | 12 | 13 | 13 | 15 | 14 | 10 | 12 |
| C18. | 18 | 10 | 15 | 11 | 10 | 8 | 10 | 9 | 14 | 11 | 11 | 11 | 12 | 12 | 13 | 13 | 14 | 12 |
| C19. | 21 | 12 | 11 | 8 | 13 | 12 | 15 | 12 | 16 | 14 | 14 | 14 | 14 | 14 | 16 | 15 | 15 | 13 |
| C20. | 16 | 13 | 14 | 12 | 18 | 13 | 17 | 13 | 16 | 15 | 15 | 15 | 15 | 15 | 17 | 16 | 16 | 14 |
| C21. | 11 | 11 | 13 | 10 | 15 | 12 | 16 | 13 | 17 | 16 | 16 | 16 | 16 | 16 | 18 | 17 | 17 | 15 |
| C22. | 19 | 12 | 13 | 11 | 17 | 13 | 18 | 14 | 18 | 17 | 17 | 17 | 17 | 17 | 19 | 18 | 18 | 16 |

Tabella 5.8: Tabella relativa al Mondiale F1 2008, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica generale, al termine della gara stessa. I valori sono ricavati dall'applicazione dell'algoritmo di Condorcet al termine di ogni gara, considerando anche le precedenti. Ogni gara rappresenta un voto, composto dagli ordini di arrivo.

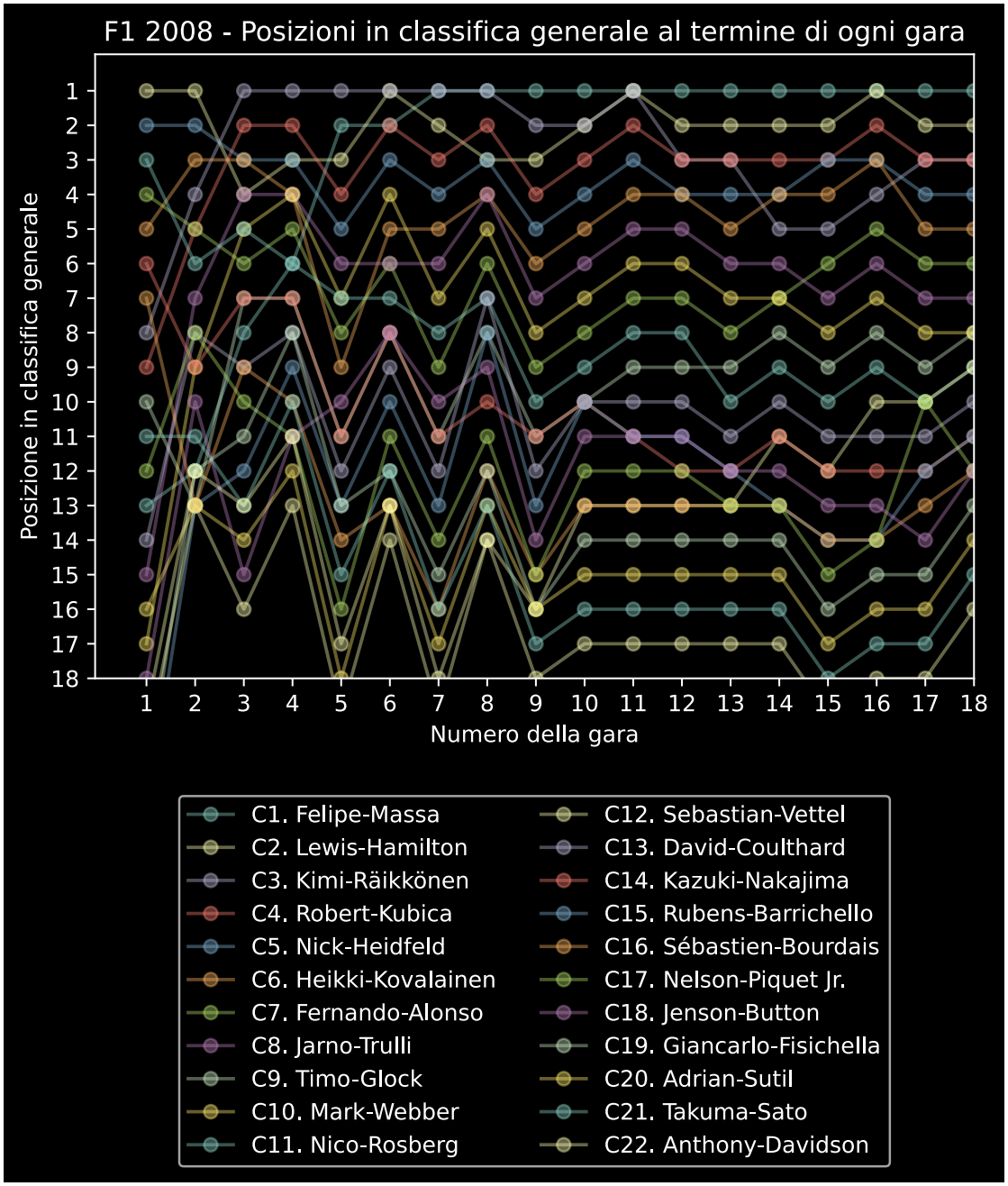


Figura 5.5: Grafico a linee in cui sono utilizzati i valori presenti nella tabella 5.8

| | G1 | G2 | G3 | G4 | G5 | G6 | G7 | G8 | G9 | G10 | G11 | G12 | G13 | G14 | G15 | G16 | G17 | G18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1. | 13 | 14 | 9 | 7 | 5 | 3 | 3 | 3 | 5 | 4 | 6 | 5 | 3 | 3 | 4 | 4 | 4 | 3 |
| C2. | 1 | 1 | 5 | 4 | 3 | 2 | 3 | 4 | 3 | 3 | 3 | 2 | 1 | 2 | 1 | 2 | 1 | 1 |
| C3. | 8 | 3 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 3 | 6 | 6 | 6 | 6 | 5 | 5 |
| C4. | 9 | 4 | 4 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 2 |
| C5. | 2 | 2 | 2 | 3 | 4 | 4 | 2 | 5 | 4 | 4 | 4 | 6 | 4 | 4 | 3 | 3 | 3 | 4 |
| C6. | 5 | 2 | 3 | 5 | 6 | 6 | 4 | 6 | 6 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 7 | 7 |
| C7. | 4 | 5 | 6 | 9 | 7 | 7 | 6 | 8 | 9 | 7 | 8 | 9 | 8 | 8 | 7 | 7 | 6 | 6 |
| C8. | 15 | 7 | 7 | 6 | 7 | 8 | 5 | 7 | 7 | 6 | 7 | 7 | 7 | 7 | 8 | 8 | 8 | 8 |
| C9. | 10 | 13 | 11 | 12 | 11 | 11 | 8 | 10 | 11 | 10 | 11 | 11 | 10 | 10 | 10 | 11 | 10 | 10 |
| C10. | 17 | 9 | 8 | 8 | 7 | 5 | 4 | 7 | 8 | 7 | 9 | 8 | 9 | 9 | 9 | 9 | 9 | 9 |
| C11. | 3 | 6 | 7 | 10 | 8 | 9 | 7 | 9 | 10 | 8 | 10 | 10 | 11 | 11 | 11 | 10 | 11 | 11 |
| C12. | 20 | 19 | 20 | 20 | 19 | 19 | 14 | 16 | 18 | 15 | 17 | 16 | 17 | 14 | 13 | 12 | 12 | 12 |
| C13. | 14 | 8 | 12 | 13 | 9 | 13 | 9 | 11 | 14 | 12 | 13 | 13 | 13 | 13 | 13 | 14 | 15 | 15 |
| C14. | 6 | 8 | 10 | 11 | 10 | 10 | 10 | 12 | 13 | 9 | 12 | 12 | 12 | 12 | 12 | 13 | 13 | 13 |
| C15. | 22 | 17 | 16 | 16 | 14 | 14 | 12 | 13 | 12 | 11 | 14 | 14 | 15 | 16 | 15 | 16 | 17 | 17 |
| C16. | 7 | 12 | 14 | 17 | 16 | 18 | 15 | 18 | 17 | 16 | 18 | 17 | 18 | 18 | 16 | 17 | 18 | 18 |
| C17. | 12 | 8 | 13 | 17 | 15 | 16 | 14 | 15 | 16 | 13 | 15 | 13 | 14 | 15 | 14 | 14 | 14 | 14 |
| C18. | 18 | 11 | 17 | 14 | 12 | 12 | 11 | 14 | 15 | 14 | 16 | 15 | 16 | 17 | 14 | 15 | 16 | 16 |
| C19. | 21 | 15 | 15 | 14 | 13 | 15 | 13 | 17 | 17 | 17 | 19 | 18 | 19 | 19 | 17 | 18 | 19 | 19 |
| C20. | 16 | 18 | 19 | 19 | 18 | 20 | 17 | 20 | 20 | 18 | 20 | 19 | 20 | 20 | 18 | 19 | 20 | 20 |
| C21. | 11 | 10 | 14 | 15 | 15 | 17 | 16 | 19 | 19 | 19 | 21 | 20 | 21 | 21 | 19 | 20 | 21 | 21 |
| C22. | 19 | 16 | 18 | 18 | 17 | 21 | 18 | 21 | 21 | 20 | 22 | 21 | 22 | 22 | 20 | 21 | 22 | 22 |

Tabella 5.9: Tabella relativa al Mondiale F1 2008, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica generale, al termine della gara stessa. I valori sono ricavati dall'applicazione dell'algoritmo di Schultze al termine di ogni gara, considerando anche le precedenti. Ogni gara rappresenta un voto, composto dagli ordini di arrivo.

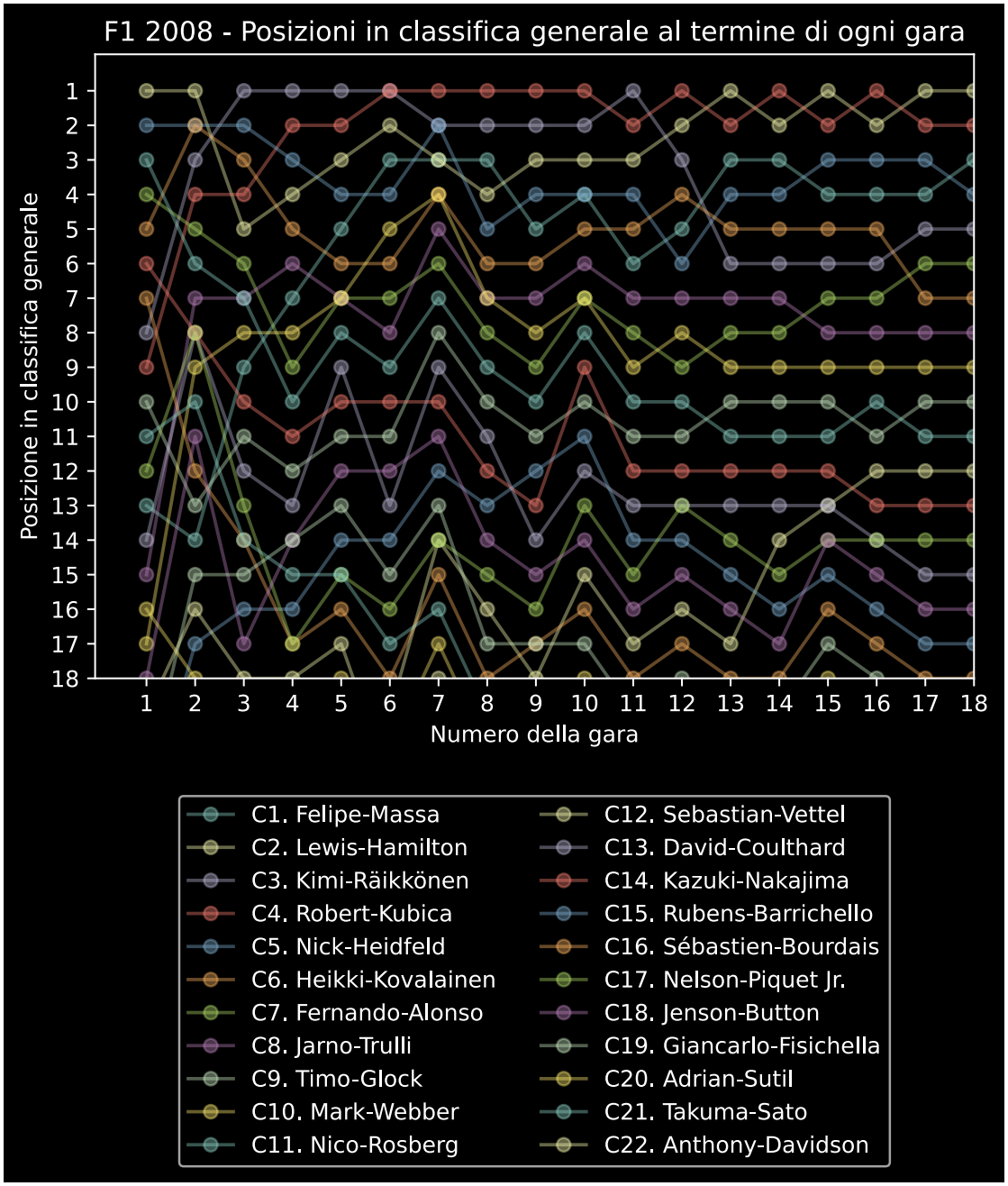


Figura 5.6: Grafico a linee in cui sono utilizzati i valori presenti nella tabella 5.9

Gli algoritmi impiegati hanno permesso di elaborare una differente distribuzione della prima classifica provvisoria rispetto alla realtà, in cui molti piloti erano

finiti fuori gara per incidenti o problemi di motore. Al pari del caso inerente al campionato 2023, si continua ad evidenziare una fase iniziale con numerosi pareggi per poi distinguerli successivamente, tuttavia i risultati di Condorcet mostrano limiti in merito alla capacità di discriminazione di un vincitore unico per ogni posizione. In tabella 5.10 vengono mostrati i valori di distanza ottenuti applicando la *norma di Frobenius* alla differenza delle matrici rappresentate nelle prime due colonne. Rispetto al caso del campionato 2023 si può notare una differenza più marcata in termini di distanze, sia nel caso di Schultze che in Condorcet. Quest'ultimo mostra una maggior differenza sia nel caso in cui viene confrontato l'insieme delle classifiche provvisorie, sia nel caso della sola classifica finale.

| Elemento 1 | Elemento 2 | Valore distanza euclidea |
|---|---|---|
| Matrice "reale" | Matrice Condorcet | 77.19 |
| Matrice "reale" | Matrice Schultze | 52.53 |
| Classifica finale "reale" | Classifica finale Condorcet | 24.6 |
| Classifica finale "reale" | Classifica finale Schultze | 12.45 |

Tabella 5.10: Campionato F1 2008 - Valori di distanza ottenuti applicando la *norma di Frobenius* alla differenza delle matrici rappresentate nelle prime due colonne.

# Capitolo 6

# Conclusioni

In questa tesi è stata presentata un'introduzione sulla *democrazia* e sullo stato attuale della *democrazia digitale*, illustrando soluzioni esistenti sia nell'ambito delle decisioni politiche che nell'ambito di contesti logistici impattanti la quotidianità della popolazione. A seguire è stata presentata la creazione di una libreria multipiattaforma in Kotlin Multiplatform, avente lo scopo di rendere disponibile uno strumento per gestire le votazioni digitali, parametrandole come ritenuto più opportuno dall'utilizzatore finale, ossia dall'organizzatore della votazione. Questa libreria è stata progettata ponendo l'accento su una tecnologia multipiattaforma per massimizzarne la diffusione su differenti sistemi, assicurando la piena compatibilità con gli stessi. Per fare ciò è stata svolta l'analisi del modello del dominio da trattare, in particolare si è reso evidente che esistono due tipologie principali di voto che caratterizzano una possibile votazione. Al fine di agevolare l'utilizzo della libreria sono stati ideati dei DSL utili a semplificare la definizione e la struttura di un pool delle votazioni, riportando per essi i diagrammi UML architetturali. Successivamente sono stati descritti i passi implementativi salienti, corredati dai test effettuati e dagli strumenti utilizzati nel contesto di *build automation*. Riguardo a questi ultimi, sono stati illustrati gli step realizzati per arrivare alla generazione e pubblicazione degli artefatti finali, compresa la documentazione a corredo. Infine viene mostrata un'estensione dei casi studio, applicando le funzionalità realizzate a dati reali relativi ai campionati F1. Per questi ultimi vengono approfonditi i

dettagli relativi alle classifiche ricalcolate utilizzando la libreria e vengono posti a confronto con i dati reali, misurando le differenze tramite un'opportuna metrica di distanza. La possibilità di ottenere dei pareggi nelle classifiche generate ha permesso di ottenere una maggior espressività sul piazzamento dei concorrenti, ma allo stesso tempo ha comportato una maggior distanza rispetto alle classifiche reali, che non riportavano la possibilità di pareggio. Le future evoluzioni della libreria possono consistere nella gestione di eventi sportivi che riportano una struttura più complessa, ad esempio l'adozione di gironi, e secondo, ma non per importanza, l'estensione degli algoritmi disponibili sia per le votazioni a singola preferenza che per quelle a lista di preferenze.

# Bibliografia


[AR19] Georg Aichholzer and Gloria Rose. *Experience with Digital Tools in Different Types of e-Participation*, page 93–140. Springer International Publishing, November 2019.

[BT17] Prabh Bedi and Neha Goel Tripathi. *Smart Water Management and eDemocracy in India*, page 259–279. Springer Singapore, 2017.

[CF19] Cristiano Castelfranchi and Rino Falcone. The problematic relationship between trust and democracy; its crisis and web dangers and promises. In Pierluigi Contucci, Andrea Omicini, Danilo Pianini, and Alina Sîrbu, editors, *The Future of Digital Democracy - An Interdisciplinary Approach*, volume 11300 of *Lecture Notes in Computer Science*, pages 62–82. Springer, 2019.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, Third Edition*. Johns Hopkins University Press, 1996.

[HKvK+19] Leonhard Hennen, Iris Korthagen, Ira van Keulen, Georg Aichholzer, Ralf Lindner, and Rasmus Ø. Nielsen. *Introduction*, page 1–8. Springer International Publishing, November 2019.

[KB22] Ilona Kozak and Andrii Berko. Three-module framework for automated software testing. In *17th IEEE International Conference on Computer Sciences and Information Technologies, CSIT 2022, Lviv, Ukraine, November 10-12, 2022*, pages 454–457. IEEE, 2022.

[KLN19] Iris Korthagen, Casper Freundlich Larsen, and Rasmus Ø. Nielsen. *Non-binding Decision-Making*, page 237–271. Springer International Publishing, November 2019.

[MM04] Margaret McGaley and Joe McCarthy. Transparency and e-voting: Democratic vs. commercial interests. In Alexander Prosser and Robert Krimmer, editors, *Electronic Voting in Europe - Technology, Law, Politics and Society, Workshop of the ESF TED Programme together with GI and OCG, July, 7th-9th, 2004, in Schloß Hofen / Bregenz, Lake of Constance, Austria, Proceedings*, volume P-47 of *LNI*, pages 153–163. GI, 2004.

[PO18] Danilo Pianini and Andrea Omicini. *Democratic Process and Digital Platforms: An Engineering Perspective*, page 83–96. Springer International Publishing, December 2018.

[RSW22] Narcyz Roztocki, Wojciech Strzelczyk, and Heinz Weistroffer. Concepts of e-democracy in an e-society, sais 2022 proceedings. 13. 2022.

[Sot20] Pedro Soto-Acosta. COVID-19 pandemic: Shifting digital transformation to a high-speed gear. *Inf. Syst. Manag.*, 37(4):260–266, 2020.

[SVBH+19] Cesar Soto-Valero, Amine Benelallam, Nicolas Harrand, Olivier Barais, and Benoit Baudry. The emergence of software diversity in maven central. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, May 2019.

[Tra] Thomas Trapanese. *Modelli e piattaforme per la democrazia digitale: analisi e confronto*. PhD thesis.

[VK17] T. M. Vinod Kumar. *State of the Art of E-Democracy for Smart Cities*, page 1–47. Springer Singapore, 2017.